salesforce

Metadata API Developer Guide

Version 48.0, Spring '20

@salesforcedocs

Last updated: June 9, 2020

CONTENTS

GETTING STARTED

CHAPTER 1 Understanding Metadata API

Use Metadata API to retrieve, deploy, create, update or delete customization information, such as custom object definitions and page layouts, for your organization. This API is intended for managing customizations and for building tools that can manage the metadata model, not the data itself. To create, retrieve, update or delete records, such as accounts or leads, use data SOAP API or REST API.

The easiest way to access the functionality in Metadata API is to use the Salesforce Extensions for Visual Studio Code or the Ant Migration Tool. Both tools are built on top of Metadata API and use the standard tools to simplify working with Metadata API.

- The Salesforce Extensions for Visual Studio Code includes tools for developing on the Salesforce platform in the lightweight, extensible VS Code editor. These tools provide features for working with development orgs (scratch orgs, sandboxes, and DE orgs), Apex, Aura components, and Visualforce.
- The Ant Migration Tool is ideal if you use a script or the command line for moving metadata between a local directory and a Salesforce org.

For more information about the Salesforce Extensions for Visual Studio Code or the Ant Migration Tool, see Salesforce Tools and Toolkits.

The underlying calls of Metadata API have been exposed for you to use directly, if you prefer to build your own client applications. This guide gives you more information about working directly with Metadata API.

You can use the Metadata API to manage setup and customization information (metadata) for your organizations. For example:

- Export the customizations in your organization as XML metadata files. See Working with the Zip File and `retrieve()`.
- Migrate configuration changes between organizations. See `deploy()` and `retrieve()`.
- Modify existing customizations in your organization using XML metadata files. See `deploy()` and `retrieve()`.
- Manage customizations in your organization programmatically. See CRUD-Based Metadata Development.

You can modify metadata in test organizations on Developer Edition or sandbox, and then deploy tested changes to production organizations on Enterprise, Unlimited, or Performance Editions. You can also create scripts to populate a new organization with your custom objects, custom fields, and other components.

SEE ALSO:

Deploying and Retrieving Metadata

CRUD-Based Metadata Development

Metadata Components and Types

Supported Salesforce Editions

To use Metadata API, your organization must use **Enterprise Edition**, **Unlimited Edition**, **Performance Edition**, or **Developer Edition**. If you are an existing Salesforce customer and want to upgrade to Enterprise, Unlimited, or Performance Edition, contact your account representative.

It is strongly recommended that you use a sandbox, which is an exact replica of your production organization. Enterprise, Unlimited, and Performance Editions come with free developer sandboxes. For more information, see `http://www.salesforce.com/platform/cloud-infrastructure/sandbox.jsp`.

Alternatively, you can use a Developer Edition org, which provides access to all of the features available with Enterprise Edition, but is limited by the number of users and the amount of storage space. A Developer Edition org isn't a copy of your production org, but it provides an environment where you can build and test your solutions without affecting your organization's data. Developer Edition accounts are available for free at `http://developer.salesforce.com/signup`.

Note: A metadata component must be visible in the org for Metadata API to act on it. Also, a user must have the API Enabled permission to have access to metadata components.

Metadata API Access for Professional Edition

ISV partners can request Metadata API access to Professional Edition orgs for apps that have passed the AppExchange Security Review. Access is granted through an API token (client ID). This special key enables the app to make Metadata API calls to customers' Professional Edition orgs.

As an ISV partner, you can request Metadata API access by following these steps.

1. Submit your app for security review. See Steps in the Security Review in the *ISVForce Guide*.
2. After your app is approved, log a case in the Partner Community in **AppExchange and Feature Requests** > **API Token Request**, and specify SOAP for the type of token.

To make calls to the Metadata API, append the API token to the CallOptions SOAP header in your calls.

Modify Metadata Through Metadata API Functions Permission

Users with the Modify Metadata Through Metadata API Functions permission can update metadata (including Apex) through Metadata API even if they don't also have the Modify All Data permission. Metadata API is used for deployments using change sets, the Ant Migration Tool, or the Salesforce CLI. Users must have the permission that enables use of the feature supported by the metadata they're trying to modify. They must also have the permission that enables their deployment tool.

The Modify Metadata Through Metadata API Functions permission doesn't impact direct customization of metadata using Setup UI pages, because those pages don't use Metadata API for updates.

Some metadata, such as Apex, executes in system context, so be careful how you delegate the Modify Metadata Through Metadata API Functions permission. Modify Metadata Through Metadata API Functions allows deployment of Apex metadata, but it doesn't allow some Apex development and debugging features that still require the Modify All Data permission.

Modify Metadata Through Metadata API Functions is enabled automatically when either the Deploy Change Sets or the Author Apex permission is selected.

Development Platforms

Metadata API supports both file-based and CRUD-based development.

File-Based Development

The declarative or file-based asynchronous Metadata API `deploy()` and `retrieve()` operations deploy or retrieve a `.zip` file that holds components in a set of folders, and a manifest file named `package.xml`. For more information, see Deploying and Retrieving

Metadata on page 16. The easiest way to access the file-based functionality is to use the Salesforce Extensions for Visual Studio Code or the Ant Migration Tool.

CRUD-Based Development

The CRUD Metadata API calls act upon the metadata components in a manner similar to the way synchronous API calls in the *enterprise WSDL* act upon objects. For more information about the enterprise WSDL, see the *SOAP API Developer Guide*.

Standards Compliance

Metadata API is implemented to comply with the following specifications:

Standard Name	Website
Simple Object Access Protocol (SOAP) 1.1	`http://www.w3.org/TR/2000/NOTE-SOAP-20000508/`
Web Service Description Language (WSDL) 1.1	`http://www.w3.org/TR/2001/NOTE-wsdl-20010315`
WS-I Basic Profile 1.1	`http://www.ws-i.org/Profiles/BasicProfile-1.1-2004-08-24.html`

Metadata API Support Policy

Salesforce supports previous versions of Metadata API. However, your new client applications should use the most recent version of the Lightning Platform Metadata API WSDL file to fully exploit the benefits of richer features and greater efficiency.

Backward Compatibility

Salesforce strives to make backward compatibility easy when using the Lightning Platform.

Each new Salesforce release consists of two components:

- A new release of platform software that resides on Salesforce systems
- A new version of the API

For example, the Spring '07 release included API version 9.0 and the Summer '07 release included API version 10.0.

We maintain support for each API version across releases of the platform software. The API is backward compatible in that an application created to work with a given API version will continue to work with that same API version in future platform software releases.

Salesforce does not guarantee that an application written against one API version will work with future API versions: Changes in method signatures and data representations are often required as we continue to enhance the API. However, we strive to keep the API consistent from version to version with minimal, if any, changes required to port applications to newer API versions.

For example, an application written using API version 9.0, which shipped with the Spring '07 release, will continue to work with API version 9.0 on the Summer '07 release, and on future releases beyond that. However, that same application might not work with API version 10.0 without modifications to the application.

API End-of-Life

Salesforce is committed to supporting each API version for a minimum of three years from the date of first release. In order to mature and improve the quality and performance of the API, versions that are more than three years old might cease to be supported.

When an API version is to be deprecated, advance notice is given at least one year before support ends. Salesforce will directly notify customers using API versions planned for deprecation.

Related Resources

The Salesforce developer website provides a full suite of developer toolkits, sample code, sample SOAP messages, community-based support, and other resources to help you with your development projects. Be sure to visit `https://developer.salesforce.com/page/Getting_Started` for more information, or visit `http://developer.salesforce.com/signup` to sign up for a free Developer Edition account.

You can visit these websites to find out more about Salesforce applications:

- Salesforce Developers provides a wealth of information for developers.
- Salesforce for information about the Salesforce application.
- Lightning Platform AppExchange for access to apps created for Salesforce.
- Salesforce.com Community for services to ensure Salesforce customer success.

CHAPTER 2 Quick Start

Use Metadata API to retrieve, deploy, create, update, or delete customizations for your org. The most common use is to migrate changes from a sandbox or testing org to your production environment. Metadata API is intended for managing customizations and for building tools that can manage the metadata model, not the data itself.

However, the underlying calls of Metadata API have been exposed for you to use directly, if you prefer to build your own client applications. This quick start gives you all the information you need to start writing applications that directly use Metadata API to manage customizations for your organization. It shows you how to get started with File-Based Development. For an example of CRUD-Based Development, see Java Sample for CRUD-Based Development with Synchronous Calls.

Prerequisites

Make sure you complete these prerequisites before you start using Metadata API.

- Create a development environment.

  It is strongly recommended that you use a sandbox, which is an exact replica of your production organization. Enterprise, Unlimited, and Performance Editions come with free developer sandboxes. For more information, see `http://www.salesforce.com/platform/cloud-infrastructure/sandbox.jsp`.

  Alternatively, you can use a Developer Edition org, which provides access to all of the features available with Enterprise Edition, but is limited by the number of users and the amount of storage space. A Developer Edition org isn't a copy of your production org, but it provides an environment where you can build and test your solutions without affecting your organization's data. Developer Edition accounts are available for free at `http://developer.salesforce.com/signup`.

- Identify a user that has the API Enabled permission and the Modify All Data or Modify Metadata Through Metadata API Functions permissions. These permissions are required to access Metadata API calls.

  Note: If a user requires access to metadata but not to data, you can enable the Modify Metadata Through Metadata API Functions permission to give the access the user needs without providing access to org data.

- Install a SOAP client. Metadata API works with current SOAP development environments, including, but not limited to, Visual Studio® .NET and the Web Service Connector (WSC).

  In this document, we provide Java examples based on WSC and JDK 6 (Java Platform Standard Edition Development Kit 6). To run the samples, first download the latest force-wsc JAR file and its dependencies (dependencies are listed on the page when you select a version) from mvnrepository.com/artifact/com.force.api/force-wsc/.

  Note: Development platforms vary in their SOAP implementations. Implementation differences in certain development platforms might prevent access to some or all of the features in Metadata API.

Step 1: Generate or Obtain the Web Service WSDLs for Your Organization

To access Metadata API calls, you need a Web Service Description Language (WSDL) file. The WSDL file defines the Web service that is available to you. Your development platform uses this WSDL to generate stub code to access the Web service it defines. You can either obtain the WSDL file from your organization's Salesforce administrator, or you can generate it yourself if you have access to the WSDL download page in the Salesforce user interface. For more information about WSDL, see `http://www.w3.org/TR/wsdl`.

Before you can access Metadata API calls, you must authenticate to use the Web service using the `login()` call, which is defined in the enterprise WSDL and the partner WSDL. Therefore, you must also obtain one of these WSDLs.

Any user with the Modify All Data or Modify Metadata Through Metadata API Functions permission can download the WSDL file to integrate and extend the Salesforce platform.

Note: If a user requires access to metadata but not to data, you can enable the Modify Metadata Through Metadata API Functions permission to give the access the user needs without providing access to org data.

The sample code in Step 3: Walk Through the Java Sample Code on page 7 uses the enterprise WSDL, though the partner WSDL works equally well.

To generate the metadata and enterprise WSDL files for your organization:

1. Log in to your Salesforce account. You must log in as an administrator or as a user who has the "Modify All Data" permission.
2. From Setup, enter *`API`* in the `Quick Find` box, then select **API**.
3. Click **Generate Metadata WSDL** and save the XML WSDL file to your file system.
4. Click **Generate Enterprise WSDL** and save the XML WSDL file to your file system.

Step 2: Import the WSDL Files Into Your Development Platform

Once you have the WSDL files, import them into your development platform so that your development environment can generate the necessary objects for use in building client Web service applications. This section provides sample instructions for WSC. For instructions about other development platforms, see your platform's product documentation.

Note: The process for importing WSDL files is identical for the metadata and enterprise WSDL files.

Instructions for Java Environments (WSC)

Java environments access the API through Java objects that serve as proxies for their server-side counterparts. Before using the API, you must first generate these objects from your organization's WSDL file.

Each SOAP client has its own tool for this process. For WSC, use the `wsdlc` utility.

Note: Before you run `wsdlc`, you must have the WSC JAR file installed on your system and referenced in your classpath. You can download the latest force-wsc JAR file and its dependencies (dependencies are listed on the page when you select a version) from mvnrepository.com/artifact/com.force.api/force-wsc/.

The basic syntax for `wsdlc` is:

```
java -classpath pathToWsc;pathToWscDependencies com.sforce.ws.tools.wsdlc
pathToWsdl/WsdlFilename pathToOutputJar/OutputJarFilename
```

For example, on Windows:

```
java -classpath force-wsc-30.0.0.jar;ST4-4.0.7.jar;antlr-runtime-3.5.jar
com.sforce.ws.tools.wsdlc metadata.wsdl metadata.jar
```

On Mac OS X and Unix, use a colon instead of a semicolon in between items in the classpath:

```
java -classpath force-wsc-30.0.0.jar:ST4-4.0.7.jar:antlr-runtime-3.5.jar
com.sforce.ws.tools.wsdlc metadata.wsdl metadata.jar
```

`wsdlc` generates a JAR file and Java source code and bytecode files for use in creating client applications. Repeat this process for the enterprise WSDL to create an enterprise.JAR file.

Step 3: Walk Through the Java Sample Code

When you have imported the WSDL files, you can build client applications that use Metadata API. The sample is a good starting point for writing your own code.

Before you run the sample, modify your project and the code to:

1. Include the WSC JAR, its dependencies, and the JAR files you generated from the WSDLs.

   Note: Although WSC has other dependencies, the following sample only requires Rhino (`js-1.7R2.jar`), which you can download from mvnrepository.com/artifact/rhino/js.

2. Update USERNAME and PASSWORD variables in the `MetadataLoginUtil.login()` method with your user name and password. If your current IP address isn't in your organization's trusted IP range, you'll need to append a security token to the password.
3. If you are using a sandbox, be sure to change the login URL.

Login Utility

Java users can use `ConnectorConfig` to connect to Enterprise, Partner, and Metadata SOAP API. `MetadataLoginUtil` creates a `ConnectorConfig` object and logs in using the Enterprise WSDL login method. Then it retrieves `sessionId` and `metadataServerUrl` to create a `ConnectorConfig` and connects to Metadata API endpoint. `ConnectorConfig` is defined in WSC.

The `MetadataLoginUtil` class abstracts the login code from the other parts of the sample, allowing portions of this code to be reused without change across different Salesforce APIs.

```
import com.sforce.soap.enterprise.EnterpriseConnection;
import com.sforce.soap.enterprise.LoginResult;
import com.sforce.soap.metadata.MetadataConnection;
import com.sforce.ws.ConnectionException;
import com.sforce.ws.ConnectorConfig;

/**
 * Login utility.
 */
public class MetadataLoginUtil {

    public static MetadataConnection login() throws ConnectionException {
        final String USERNAME = "user@company.com";
        // This is only a sample. Hard coding passwords in source files is a bad practice.
        final String PASSWORD = "password";
        final String URL = "https://login.salesforce.com/services/Soap/c/48.0";
        final LoginResult loginResult = loginToSalesforce(USERNAME, PASSWORD, URL);
        return createMetadataConnection(loginResult);
    }

    private static MetadataConnection createMetadataConnection(
            final LoginResult loginResult) throws ConnectionException {
        final ConnectorConfig config = new ConnectorConfig();
        config.setServiceEndpoint(loginResult.getMetadataServerUrl());
        config.setSessionId(loginResult.getSessionId());
        return new MetadataConnection(config);
    }

    private static LoginResult loginToSalesforce(
            final String username,
            final String password,
            final String loginUrl) throws ConnectionException {
        final ConnectorConfig config = new ConnectorConfig();
        config.setAuthEndpoint(loginUrl);
        config.setServiceEndpoint(loginUrl);
        config.setManualLogin(true);
        return (new EnterpriseConnection(config)).login(username, password);
    }
}
```

Note: This example uses user and password authentication to obtain a session ID, which is then used for making calls to Metadata API. Alternatively, you can use OAuth authentication. After you athenticate with OAuth to Salesforce, pass the returned access token instead of the session ID. For example, pass the access token to the `setSessionId()` call on `ConnectorConfig`. To learn how to use OAuth authentication in Salesforce, see Authenticating Apps with OAuth in the Salesforce Help.

Java Sample Code for File-Based Development

The sample code logs in using the login utility. Then it displays a menu with retrieve, deploy, and exit.

The `retrieve()` and `deploy()` calls both operate on a .zip file named `components.zip`. The `retrieve()` call retrieves components from your organization into `components.zip`, and the `deploy()` call deploys the components in `components.zip` to your organization. If you save the sample to your computer and execute it, run the retrieve option first so that you have a `components.zip` file that you can subsequently deploy. After a retrieve call, the sample calls `checkRetrieveStatus()` in a loop until the operation is completed. Similarly, after a deploy call, the sample checks `checkDeployStatus()` in a loop until the operation is completed.

The `retrieve()` call uses a manifest file to determine the components to retrieve from your organization. A sample `package.xml` manifest file follows. For more details on the manifest file structure, see Deploying and Retrieving Metadata with the Zip File. For this sample, the manifest file retrieves all custom objects, custom tabs, and page layouts.

```
<?xml version="1.0" encoding="UTF-8"?>
<Package xmlns="http://soap.sforce.com/2006/04/metadata">
    <types>
        <members>*</members>
        <name>CustomObject</name>
    </types>
    <types>
        <members>*</members>
        <name>CustomTab</name>
    </types>
    <types>
        <members>*</members>
        <name>Layout</name>
    </types>
    <version>48.0</version>
</Package>
```

Note the error handling code that follows each API call.

Note: This sample requires API version 34.0 or later.

```
import java.io.*;
import java.nio.channels.Channels;
import java.nio.channels.FileChannel;
import java.nio.channels.ReadableByteChannel;
import java.rmi.RemoteException;
import java.util.*;

import javax.xml.parsers.*;

import org.w3c.dom.*;
import org.xml.sax.SAXException;

import com.sforce.soap.metadata.*;

/**
 * Sample that logs in and shows a menu of retrieve and deploy metadata options.
 */
public class FileBasedDeployAndRetrieve {

    private MetadataConnection metadataConnection;

    private static final String ZIP_FILE = "components.zip";

    // manifest file that controls which components get retrieved
    private static final String MANIFEST_FILE = "package.xml";

    private static final double API_VERSION = 29.0;

    // one second in milliseconds
    private static final long ONE_SECOND = 1000;

    // maximum number of attempts to deploy the zip file
    private static final int MAX_NUM_POLL_REQUESTS = 50;

    private BufferedReader reader = new BufferedReader(new InputStreamReader(System.in));


    public static void main(String[] args) throws Exception {
        FileBasedDeployAndRetrieve sample = new FileBasedDeployAndRetrieve();
        sample.run();
    }

```

```
public FileBasedDeployAndRetrieve() {
}

private void run() throws Exception {
    this.metadataConnection = MetadataLoginUtil.login();

    // Show the options to retrieve or deploy until user exits
    String choice = getUsersChoice();
    while (choice != null && !choice.equals("99")) {
        if (choice.equals("1")) {
            retrieveZip();
        } else if (choice.equals("2")) {
            deployZip();
        } else {
            break;
        }
        // show the options again
        choice = getUsersChoice();
    }
}

/*
 * Utility method to present options to retrieve or deploy.
 */
private String getUsersChoice() throws IOException {
    System.out.println(" 1: Retrieve");
    System.out.println(" 2: Deploy");
    System.out.println("99: Exit");
    System.out.println();
    System.out.print("Enter 1 to retrieve, 2 to deploy, or 99 to exit: ");
    // wait for the user input.
    String choice = reader.readLine();
    return choice != null ? choice.trim() : "";
}

private void deployZip() throws Exception {
    byte zipBytes[] = readZipFile();
    DeployOptions deployOptions = new DeployOptions();
    deployOptions.setPerformRetrieve(false);
    deployOptions.setRollbackOnError(true);
    AsyncResult asyncResult = metadataConnection.deploy(zipBytes, deployOptions);
    DeployResult result = waitForDeployCompletion(asyncResult.getId());
    if (!result.isSuccess()) {
        printErrors(result, "Final list of failures:\n");
        throw new Exception("The files were not successfully deployed");
    }
    System.out.println("The file " + ZIP_FILE + " was successfully deployed\n");
}

/*
* Read the zip file contents into a byte array.
*/
private byte[] readZipFile() throws Exception {
    byte[] result = null;
```

```
        // We assume here that you have a deploy.zip file.
        // See the retrieve sample for how to retrieve a zip file.
        File zipFile = new File(ZIP_FILE);
        if (!zipFile.exists() || !zipFile.isFile()) {
            throw new Exception("Cannot find the zip file for deploy() on path:"
                + zipFile.getAbsolutePath());
        }

        FileInputStream fileInputStream = new FileInputStream(zipFile);
        try {
            ByteArrayOutputStream bos = new ByteArrayOutputStream();
            byte[] buffer = new byte[4096];
            int bytesRead = 0;
            while (-1 != (bytesRead = fileInputStream.read(buffer))) {
                bos.write(buffer, 0, bytesRead);
            }

            result = bos.toByteArray();
        } finally {
            fileInputStream.close();
        }
        return result;
    }

    /*
    * Print out any errors, if any, related to the deploy.
    * @param result - DeployResult
    */
    private void printErrors(DeployResult result, String messageHeader) {
        DeployDetails details = result.getDetails();
        StringBuilder stringBuilder = new StringBuilder();
        if (details != null) {
            DeployMessage[] componentFailures = details.getComponentFailures();
            for (DeployMessage failure : componentFailures) {
                String loc = "(" + failure.getLineNumber() + ", " +
failure.getColumnNumber();
                if (loc.length() == 0 &&
!failure.getFileName().equals(failure.getFullName()))
                {
                    loc = "(" + failure.getFullName() + ")";
                }
                stringBuilder.append(failure.getFileName() + loc + ":"
                    + failure.getProblem()).append('\n');
            }
            RunTestsResult rtr = details.getRunTestResult();
            if (rtr.getFailures() != null) {
                for (RunTestFailure failure : rtr.getFailures()) {
                    String n = (failure.getNamespace() == null ? "" :
                        (failure.getNamespace() + ".")) + failure.getName();
                    stringBuilder.append("Test failure, method: " + n + "." +
                            failure.getMethodName() + " -- " + failure.getMessage() +
                            " stack " + failure.getStackTrace() + "\n\n");
                }
            }
            if (rtr.getCodeCoverageWarnings() != null) {
                for (CodeCoverageWarning ccw : rtr.getCodeCoverageWarnings()) {
                    stringBuilder.append("Code coverage issue");
                    if (ccw.getName() != null) {
                        String n = (ccw.getNamespace() == null ? "" :
                        (ccw.getNamespace() + ".")) + ccw.getName();
                        stringBuilder.append(", class: " + n);
                    }
                    stringBuilder.append(" -- " + ccw.getMessage() + "\n");
                }
            }
        }
        if (stringBuilder.length() > 0) {
            stringBuilder.insert(0, messageHeader);
            System.out.println(stringBuilder.toString());
        }
    }


    private void retrieveZip() throws Exception {
        RetrieveRequest retrieveRequest = new RetrieveRequest();
        // The version in package.xml overrides the version in RetrieveRequest
        retrieveRequest.setApiVersion(API_VERSION);
        setUnpackaged(retrieveRequest);

        AsyncResult asyncResult = metadataConnection.retrieve(retrieveRequest);
        RetrieveResult result = waitForRetrieveCompletion(asyncResult);

        if (result.getStatus() == RetrieveStatus.Failed) {
            throw new Exception(result.getErrorStatusCode() + " msg: " +
                    result.getErrorMessage());
        } else if (result.getStatus() == RetrieveStatus.Succeeded) {
         // Print out any warning messages
         StringBuilder stringBuilder = new StringBuilder();
         if (result.getMessages() != null) {
             for (RetrieveMessage rm : result.getMessages()) {
                 stringBuilder.append(rm.getFileName() + " - " + rm.getProblem() + "\n");

             }
         }
         if (stringBuilder.length() > 0) {
             System.out.println("Retrieve warnings:\n" + stringBuilder);
         }

         System.out.println("Writing results to zip file");
         File resultsFile = new File(ZIP_FILE);
         FileOutputStream os = new FileOutputStream(resultsFile);

         try {
             os.write(result.getZipFile());
         } finally {
             os.close();
         }
        }
    }

    private DeployResult waitForDeployCompletion(String asyncResultId) throws Exception {

        int poll = 0;
        long waitTimeMilliSecs = ONE_SECOND;
        DeployResult deployResult;
        boolean fetchDetails;
        do {
            Thread.sleep(waitTimeMilliSecs);
            // double the wait time for the next iteration

            waitTimeMilliSecs *= 2;
            if (poll++ > MAX_NUM_POLL_REQUESTS) {
                throw new Exception(
                    "Request timed out. If this is a large set of metadata components, "
+

                    "ensure that MAX_NUM_POLL_REQUESTS is sufficient.");
            }
            // Fetch in-progress details once for every 3 polls
            fetchDetails = (poll % 3 == 0);

          deployResult = metadataConnection.checkDeployStatus(asyncResultId, fetchDetails);

            System.out.println("Status is: " + deployResult.getStatus());
            if (!deployResult.isDone() && fetchDetails) {
                printErrors(deployResult, "Failures for deployment in progress:\n");
            }
        }
        while (!deployResult.isDone());

        if (!deployResult.isSuccess() && deployResult.getErrorStatusCode() != null) {
            throw new Exception(deployResult.getErrorStatusCode() + " msg: " +
                    deployResult.getErrorMessage());
        }

        if (!fetchDetails) {
            // Get the final result with details if we didn't do it in the last attempt.
            deployResult = metadataConnection.checkDeployStatus(asyncResultId, true);
        }

        return deployResult;
    }

    private RetrieveResult waitForRetrieveCompletion(AsyncResult asyncResult) throws
Exception {
     // Wait for the retrieve to complete
        int poll = 0;
        long waitTimeMilliSecs = ONE_SECOND;
        String asyncResultId = asyncResult.getId();
        RetrieveResult result = null;
        do {
            Thread.sleep(waitTimeMilliSecs);
            // Double the wait time for the next iteration
            waitTimeMilliSecs *= 2;
            if (poll++ > MAX_NUM_POLL_REQUESTS) {
                throw new Exception("Request timed out.  If this is a large set " +
                "of metadata components, check that the time allowed " +
                "by MAX_NUM_POLL_REQUESTS is sufficient.");
            }
            result = metadataConnection.checkRetrieveStatus(
                    asyncResultId, true);
            System.out.println("Retrieve Status: " + result.getStatus());
        } while (!result.isDone());

        return result;
    }

    private void setUnpackaged(RetrieveRequest request) throws Exception {
        // Edit the path, if necessary, if your package.xml file is located elsewhere
        File unpackedManifest = new File(MANIFEST_FILE);
        System.out.println("Manifest file: " + unpackedManifest.getAbsolutePath());

        if (!unpackedManifest.exists() || !unpackedManifest.isFile()) {
            throw new Exception("Should provide a valid retrieve manifest " +
                "for unpackaged content. Looking for " +
                unpackedManifest.getAbsolutePath());
        }

        // Note that we use the fully quualified class name because
        // of a collision with the java.lang.Package class
        com.sforce.soap.metadata.Package p = parsePackageManifest(unpackedManifest);
        request.setUnpackaged(p);
    }

    private com.sforce.soap.metadata.Package parsePackageManifest(File file)
            throws ParserConfigurationException, IOException, SAXException {
        com.sforce.soap.metadata.Package packageManifest = null;
        List<PackageTypeMembers> listPackageTypes = new ArrayList<PackageTypeMembers>();
        DocumentBuilder db =
                DocumentBuilderFactory.newInstance().newDocumentBuilder();
        InputStream inputStream = new FileInputStream(file);
        Element d = db.parse(inputStream).getDocumentElement();
        for (Node c = d.getFirstChild(); c != null; c = c.getNextSibling()) {
            if (c instanceof Element) {
                Element ce = (Element) c;
                NodeList nodeList = ce.getElementsByTagName("name");
                if (nodeList.getLength() == 0) {
                    continue;
                }
                String name = nodeList.item(0).getTextContent();
                NodeList m = ce.getElementsByTagName("members");
                List<String> members = new ArrayList<String>();
                for (int i = 0; i < m.getLength(); i++) {
                    Node mm = m.item(i);
                    members.add(mm.getTextContent());
                }
                PackageTypeMembers packageTypes = new PackageTypeMembers();
                packageTypes.setName(name);
                packageTypes.setMembers(members.toArray(new String[members.size()]));
                listPackageTypes.add(packageTypes);
            }
        }
        packageManifest = new com.sforce.soap.metadata.Package();
        PackageTypeMembers[] packageTypesArray =
                new PackageTypeMembers[listPackageTypes.size()];
        packageManifest.setTypes(listPackageTypes.toArray(packageTypesArray));
        packageManifest.setVersion(API_VERSION + "");
        return packageManifest;
    }
}
```

USING METADATA API

CHAPTER 3 Deploying and Retrieving Metadata

Use the `deploy()` and `retrieve()` calls to move metadata (XML files) between a Salesforce organization and a local file system. Once you retrieve your XML files into a file system, you can manage changes in a source-code control system, copy and paste code or setup configurations, diff changes to components, and perform many other file-based development operations. At any time you can deploy those changes to another Salesforce organization.

Note: The Ant Migration Tool uses the `deploy()` and `retrieve()` calls to move metadata. If you use these tools, interaction with Metadata API is seamless and invisible. Therefore, most developers will find it much easier to use these tools than write code that calls `deploy()` and `retrieve()` directly.

Data in XML files is formatted using the English (United States) locale. This ensures that fields that depend on locale, such as date fields, are interpreted consistently during data migrations between organizations using different languages. Organizations can support multiple languages for presentation to their users.

The `deploy()` and `retrieve()` calls are used primarily for the following development scenarios:

- Development of a custom application (or customization) in a sandbox organization. After development and testing is completed, the application or customization is then deployed into a production organization using Metadata API.
- Team development of an application in a Developer Edition organization. After development and testing is completed, you can then distribute the application via Lightning Platform AppExchange.

SEE ALSO:

Metadata Components and Types

Unsupported Metadata Types

Deploying and Retrieving Metadata with the Zip File

The `deploy()` and `retrieve()` calls are used to deploy and retrieve a .zip file. Within the .zip file is a project manifest `(package.xml)` that lists what to retrieve or deploy, and one or more XML components that are organized into folders.

Note: A component is an instance of a metadata type. For example, `CustomObject` is a metadata type for custom objects, and the `MyCustomObject__c` component is an instance of a custom object.

The files that are retrieved or deployed in a .zip file might be unpackaged components that reside in your org (such as *standard objects*) or packaged components that reside within named packages.

Note: You can deploy or retrieve up to 10,000 files at once. AppExchange packages use different limits. In API version 43.0 and 44.0, AppExchange packages can contain up to 12,500 files. In API version 45.0, AppExchange packages can contain up to 17,500 files. In API version 46.0, AppExchange packages can contain up to 22,000 files. In API version 47.0 and later, AppExchange packages can contain up to 30,000 files. The maximum size of the deployed or retrieved .zip file is 39 MB. If the files are uncompressed in an unzipped folder, the size limit is 400 MB.

- If using the Ant Migration Tool to deploy an unzipped folder, all files in the folder are compressed first. The maximum size of uncompressed components in an unzipped folder is 400 MB or less depending on the compression ratio. If the files have a high compression ratio, you can migrate a total of approximately 400 MB because the compressed size would be under 39 MB. However, if the components can't be compressed much, like binary static resources, you can migrate less than 400 MB.
- Metadata API base-64 encodes components after they're compressed. The resulting .zip file can't exceed 50 MB, which is the limit for SOAP messages. Base-64 encoding increases the size of the payload, so your compressed payload can't exceed approximately 39 MB before encoding.
- You can perform a `retrieve()` call for a big object only if its index is defined. If a big object is created in Setup and doesn't yet have an index defined, you can't retrieve it.

Every .zip file contains a project manifest, a file that's named `package.xml`, and a set of directories that contain the components. The manifest file defines the components that you're trying to retrieve or deploy in the .zip file. The manifest also defines the API version that's used for the deployment or retrieval.

Note: You can edit the project manifest, but be careful if you modify the list of components it contains. When you deploy or retrieve components, Metadata API references the components listed in the manifest, not the directories in the .zip file.

The following is a sample `package.xml` file. You can retrieve an individual component for a metadata type by specifying its `fullName` field value in a `members` element. You can also retrieve all components of a metadata type by using `<members>*</members>`.

```
<?xml version="1.0" encoding="UTF-8"?>
<Package xmlns="http://soap.sforce.com/2006/04/metadata">
    <types>
        <members>MyCustomObject__c</members>
        <name>CustomObject</name>
    </types>
    <types>
        <members>*</members>
        <name>CustomTab</name>
    </types>
    <types>
        <members>Standard</members>
        <name>Profile</name>
    </types>
    <version>48.0</version>
</Package>
```

The following elements can be defined in `package.xml`.

- `<fullName>` contains the name of the server-side package. If no `<fullName>` exists, the `package.xml` defines a client-side `unpackaged` package.
- `<types>` contains the name of the metadata type (for example, `CustomObject`) and the named members (for example, `myCustomObject__c`) to be retrieved or deployed. You can add multiple `<types>` elements in a manifest file.
- `<members>` contains the `fullName` of the component, for example `MyCustomObject__c`. The `listMetadata()` call is useful for determining the `fullName` for components of a particular metadata type if you want to retrieve an individual component. For many metadata types, you can replace the value in `members` with the wildcard character `*` (asterisk) instead of listing each member separately. See the reference topic for a specific type to determine whether that type supports wildcards.

  Note: You specify Security in the `<members>` element and Settings in the name element when retrieving the SecuritySettings component type.

- `<name>` contains the metadata type, for example `CustomObject` or `Profile`. There is one name defined for each metadata type in the directory. Any metadata type that extends Metadata is a valid value. The name that's entered must match a metadata type that's defined in the Metadata API WSDL. See Metadata Types for a list.
- `<version>` is the API version number that's used when the .zip file is deployed or retrieved. Currently the valid value is `48.0`.

For more sample `package.xml` manifest files that show you how to work with different subsets of metadata, see Sample `package.xml` Manifest Files.

To delete components, see Deleting Components from an Organization.

SEE ALSO:

Metadata Types

Does a Retrieve Job Have a Status of Pending?

If you initiate several concurrent retrieve operations for a single org, Metadata API automatically puts some of those jobs in a queue, if that becomes necessary for service protection. If a retrieve job has a status of `Pending`, it's in the queue. When one of the active retrieve jobs completes, Metadata API takes a pending job from the queue and activates it. If a retrieve job has a status of `InProgress`, it's active. The process repeats until the job queue is cleared.

For more information, see Metadata Limits in the *Salesforce Developer Limits and Allocations Quick Reference*.

Sample **`package.xml`** Manifest Files

This section includes sample `package.xml` manifest files that show you how to work with different subsets of metadata. A manifest file can include multiple `<types>` elements so you could combine the individual samples into one `package.xml` manifest file if you want to work with all the metadata in one batch. For more information about the structure of a manifest file, see Deploying and Retrieving Metadata with the Zip File. The following samples are listed:

- Standard Objects
- All Custom Objects
- Standard Picklist Fields
- Custom and Standard Fields
- List Views for Standard Objects
- Packages
- Security Settings
- Assignment Rules, Auto-response Rules, Escalation Rules
- Sharing Rules
- Managed Component Access

Standard Objects

This sample `package.xml` manifest file illustrates how to work with the standard Account object. Retrieving or deploying a standard object includes all custom and standard fields except for standard fields that aren't customizable. All custom fields are supported. Only standard fields that you can customize are supported, that is, standard fields to which you can add help text or enable history tracking or Chatter feed tracking. Other standard fields aren't supported, including system fields (such as `CreatedById` or `LastModifiedDate`) and autonumber fields.

```
<?xml version="1.0" encoding="UTF-8"?>
<Package xmlns="http://soap.sforce.com/2006/04/metadata">
    <types>
        <members>Account</members>
        <name>CustomObject</name>
    </types>
    <version>48.0</version>
</Package>
```

Note how you work with the standard Account object by specifying it as a member of a CustomObject type. However, you cannot use an asterisk wildcard to work with all standard objects; each standard object must be specified by name.

All Custom Objects

This sample `package.xml` manifest file illustrates how to work with all custom objects.

```
<?xml version="1.0" encoding="UTF-8"?>
<Package xmlns="http://soap.sforce.com/2006/04/metadata">
    <types>
        <members>*</members>
        <name>CustomObject</name>
    </types>
    <version>48.0</version>
</Package>
```

This manifest file can be used to retrieve or deploy all custom objects. This does not include all standard objects.

Standard Picklist Fields

In API version 38.0 and later, the StandardValueSet type represents standard picklists. Picklists are no longer represented by fields as in earlier versions. This sample `package.xml` represents the `Industry` standard picklist as a StandardValueSet type.

```
<?xml version="1.0" encoding="UTF-8"?>
<Package xmlns="http://soap.sforce.com/2006/04/metadata">
    <types>
        <members>Industry</members>
        <name>StandardValueSet</name>
    </types>
    <version>48.0</version>
</Package>
```

Note: The name of a standard value set is case-sensitive.

The `Industry` standard value set corresponds to the `Account.Industry` or `Lead.Industry` field in API version 37.0 and earlier. This example shows a `package.xml` sample for the `Account.Industry` picklist.

```
<?xml version="1.0" encoding="UTF-8"?>
<Package xmlns="http://soap.sforce.com/2006/04/metadata">
    <types>
        <members>Account.Industry</members>
        <name>CustomField</name>
    </types>
    <version>37.0</version>
</Package>
```

Note: The name of a picklist field is case-sensitive.

Note the ***objectName.picklistField*** syntax in the `<members>` field where *objectName* is the name of the object, such as `Account`, and *picklistField* is the name of the standard picklist field, such as `Industry`.

This next `package.xml` sample represents opportunity team roles in API version 38.0 and later. Specify opportunity team roles as a `SalesTeamRole` standard value set. Opportunity team roles have the same picklist values as the account team roles.

```
<?xml version="1.0" encoding="UTF-8"?>
<Package xmlns="http://soap.sforce.com/2006/04/metadata">
    <types>
        <members>SalesTeamRole</members>
        <name>StandardValueSet</name>
    </types>
    <version>48.0</version>
</Package>
```

The `SalesTeamRole` standard value set corresponds to one of these field names in API version 37.0 and earlier: `OpportunityTeamMember.TeamMemberRole`, `UserAccountTeamMember.TeamMemberRole`, `UserTeamMember.TeamMemberRole`, and `AccountTeamMember.TeamMemberRole`. Opportunity team roles are represented in this sample `package.xml` as the `OpportunityTeamMember.TeamMemberRole` field.

```
<?xml version="1.0" encoding="UTF-8"?>
<Package xmlns="http://soap.sforce.com/2006/04/metadata">
    <types>
        <members>OpportunityTeamMember.TeamMemberRole</members>
        <name>CustomField</name>
    </types>
    <version>37.0</version>
</Package>
```

To learn about the names of standard value sets and how they map to picklist field names, see StandardValueSet Names and Standard Picklist Fields.

Custom and Standard Fields

This sample `package.xml` manifest file illustrates how to work with custom fields in custom and standard objects and standard fields in a standard object.

```
<?xml version="1.0" encoding="UTF-8"?>
<Package xmlns="http://soap.sforce.com/2006/04/metadata">
    <types>
        <members>MyCustomObject__c.MyCustomField__c</members>
        <name>CustomField</name>
    </types>
    <types>
        <members>Account.SLA__c</members>
        <members>Account.Phone</members>
        <name>CustomField</name>
    </types>
    <version>48.0</version>
</Package>
```

Note the ***objectName.field*** syntax in the `<members>` field where *objectName* is the name of the object, such as Account, and *field* is the name of the custom or standard field, such as an `SLA` picklist field representing a service-level agreement option. The `MyCustomField` custom field in the MyCustomObject custom object is uniquely identified by its full name, `MyCustomObject__c.MyCustomField__c`. Similarly, the `Phone` standard field in the Account standard object is uniquely identified by its full name, `Account.Phone`.

All custom fields are supported. Only standard fields that you can customize are supported, that is, standard fields to which you can add help text or enable history tracking or Chatter feed tracking. Other standard fields aren't supported, including system fields (such as `CreatedById` or `LastModifiedDate`) and autonumber fields.

List Views for Standard Objects

The easiest way to retrieve list views for a standard object is to retrieve the object. The list views are included in the retrieved component. See Standard Objects on page 18.

You can also work with individual list views if you do not want to retrieve all the details for the object. This sample `package.xml` manifest file illustrates how to work with a list view for the standard Account object.

```
<?xml version="1.0" encoding="UTF-8"?>
<Package xmlns="http://soap.sforce.com/2006/04/metadata">
    <types>
        <members>Account.AccountTeam</members>
        <name>ListView</name>
    </types>
    <version>48.0</version>
</Package>
```

Note the ***objectName.listViewUniqueName*** syntax in the `<members>` field where *objectName* is the name of the object, such as Account, and *listViewUniqueName* is the `View Unique Name` for the list view. If you retrieve this list view, the component is stored in `objects/Account.object`.

Packages

To retrieve a package, set the name of the package in the `packageNames` field in `RetrieveRequest` when you call `retrieve()`. The `package.xml` manifest file is automatically populated in the retrieved `.zip` file. The `<fullName>` element in `package.xml` contains the name of the retrieved package.

If you use an asterisk wildcard in a `<members>` element to retrieve all the components of a particular metadata type, the retrieved contents do not include components in managed packages. For more information about managed packages, see the ISVforce Guide.

The easiest way to retrieve a component in a managed package is to retrieve the complete package by setting the name of the package in the `packageNames` field in `RetrieveRequest`, as described above. The following sample `package.xml` manifest file illustrates an alternative to retrieve an individual component in a package.

```
<?xml version="1.0" encoding="UTF-8"?>
<Package xmlns="http://soap.sforce.com/2006/04/metadata">
    <types>
        <members>myns__MyCustomObject__c</members>
        <name>CustomObject</name>
    </types>
    <version>48.0</version>
</Package>
```

Note the ***namespacePrefix*__*objectName*** syntax in the `<members>` field where *`namespacePrefix`* is the namespace prefix of the package and *`objectName`* is the name of the object. A namespace prefix is a 1 to 15-character alphanumeric identifier that distinguishes your package and its contents from other publishers' packages. For more information, see "Register a Namespace Prefix" in the Salesforce Help.

Security Settings

This sample `package.xml` manifest file illustrates how to work with an organization's security settings. You specify Security in the `<members>` element and Settings in the name element when retrieving the SecuritySettings component type.

```
<?xml version="1.0" encoding="UTF-8"?>
<Package xmlns="http://soap.sforce.com/2006/04/metadata">
    <types>
        <members>Security</members>
        <name>Settings</name>
    </types>
    <version>48.0</version>
</Package>
```

Assignment Rules, Auto-response Rules, Escalation Rules

Assignment rules, auto-response rules and escalation rules use different `package.xml` type names to access sets of rules or individual rules for object types. For example, the following sample `package.xml` manifest file illustrates how to access an organization's assignment rules for just Cases and Leads.

```
<?xml version="1.0" encoding="UTF-8"?>
<Package xmlns="http://soap.sforce.com/2006/04/metadata">
    <types>
        <members>Case</members>
        <members>Lead</members>
        <name>AssignmentRules</name>
    </types>
    <version>48.0</version>
</Package>
```

The following sample `package.xml` manifest file illustrates how to access just the "samplerule" Case assignment rule and the "newrule" Lead assignment rule. Notice that the type name is `AssignmentRule` and not `AssignmentRules`.

```
<?xml version="1.0" encoding="UTF-8"?>
<Package xmlns="http://soap.sforce.com/2006/04/metadata">
    <types>
        <members>Case.samplerule</members>
        <members>Lead.newrule</members>
        <name>AssignmentRule</name>
    </types>
    <version>48.0</version>
</Package>
```

Similarly, for accessing individual auto-response rules and escalation rules, use `AutoResponseRule` and `EscalationRule` instead of `AutoResponseRules` and `EscalationRules`.

Sharing Rules

In API version 33.0 and later, you can retrieve and deploy sharing rules for all standard and custom objects. This sample `package.xml` manifest file illustrates how to work with an organization's sharing rules, which includes retrieving a specific criteria-based sharing rule for the lead object, retrieving all ownership-based sharing rules for all objects, and retrieving all territory-based sharing rules for the account object.

```
<?xml version="1.0" encoding="UTF-8"?>
<Package xmlns="http://soap.sforce.com/2006/04/metadata">
    <types>
        <members>Lead.testShareRule</members>
         <name>SharingCriteriaRule</name>
    </types>
    <types>
        <members>*</members>
        <name>SharingOwnerRule</name>
    </types>
    <types>
        <members>Account.*</members>
        <name>SharingTerritoryRule</name>
    </types>
    <version>33.0</version>
</Package>
```

Managed Component Access

In API version 29.0 and later, you can retrieve and deploy access settings for the following managed components in profiles and permission sets:

- Apex classes
- Apps
- Custom field permissions
- Custom object permissions
- Custom tab settings
- External data sources
- Record types
- Visualforce pages

When retrieving and deploying managed component permissions, specify the namespace followed by two underscores. Wildcards are not supported.

For example, let's say you install a managed package with the namespace `MyNamespace` and the custom object `JobRequest__c`. To set object permissions for `JobRequest__c` in the package to the custom profile MyProfile, you would add the following to the `.profile` file.

To deploy:

```
<objectPermissions>
    <allowCreate>true</allowCreate>
    <allowDelete>true</allowDelete>
    <allowEdit>true</allowEdit>
    <allowRead>true</allowRead>
    <viewAllRecords>false</viewAllRecords>
    <modifyAllRecords>false</modifyAllRecords>
    <object>MyNamespace__JobRequest__c</object>
</objectPermissions>
```

To retrieve:

```
<types>
    <members>MyNamespace__JobRequest__c</members>
    <name>CustomObject</name>
</types>
<types>
    <members>MyProfile</members>
    <name>Profile</name>
</types>
```

When retrieving permission sets and profiles, make sure that you also retrieve any components that are related to the permissions and settings. For example, when retrieving app visibilities, you must also retrieve the associated app, and when retrieving object or field permissions, you must also retrieve the associated object.

Running Tests in a Deployment

Default Test Execution in Production

When no test level is specified in the deployment options, the default test execution behavior depends on the contents of your deployment package. When deploying to production, all tests, except those that originate from managed packages, are executed if your deployment package contains Apex classes or triggers. If your package doesn't contain Apex components, no tests are run by default.

In API version 33.0 and earlier, tests were run for components that required tests, such as custom objects, and not only for Apex components. For example, if your package contains a custom object, all tests are run in API version 33.0 and earlier. In contrast, starting with API version 34.0, no tests are run for this package. The API version corresponds to the version of your API client or the version of the tool you're using (Ant Migration Tool).

You can run tests for a deployment of non-Apex components. You can override the default test execution behavior by setting the test level in your deployment options. Test levels are enforced regardless of the types of components present in your deployment package. We recommend that you run all local tests in your development environment, such as sandbox, before deploying to production. Running tests in your development environment reduces the number of tests needed to run in a production deployment.

Default Test Execution in Production for API Version 33.0 and Earlier

For deployment to a production organization, all local tests in your organization are run by default. Tests that originate from installed managed packages aren't run by default. If any test fails, the entire deployment is rolled back.

If the deployment includes components for the following metadata types, all local tests are run.

- ApexClass
- ApexComponent
- ApexPage
- ApexTrigger
- ArticleType
- BaseSharingRule

- CriteriaBasedSharingRule
- CustomField
- CustomObject
- DataCategoryGroup
- Flow
- InstalledPackage
- NamedFilter
- OwnerSharingRule
- PermissionSet
- Profile
- Queue
- RecordType
- RemoteSiteSetting
- Role
- SharingReason
- Territory
- Validation Rules
- Workflow

For example, no tests are run for the following deployments:

- 1 CustomApplication component
- 100 Report components and 40 Dashboard components

But all local tests are run for any of the following example deployments, because they include at least one component from the list above:

- 1 CustomField component
- 1 ApexComponent component and 1 ApexClass component
- 5 CustomField components and 1 ApexPage component
- 100 Report components, 40 Dashboard components, and 1 CustomField component

SEE ALSO:

deploy()

Running a Subset of Tests in a Deployment

Test levels enable you to have more control over which tests are run in a deployment. To shorten deployment time to production, run a subset of tests when deploying Apex components. The default test execution behavior in production has also changed. By default, if no test level is specified, no tests are executed, unless your deployment package contains Apex classes or triggers.

If the code coverage of an Apex component in the deployment is less than 75%, the deployment fails. If one of the specified tests fails, the deployment also fails. We recommend that you test your deployment in sandbox first to ensure that the specified tests cover each component sufficiently. Even if your organization's overall code coverage is 75% or more, the individual coverage of the Apex components being deployed can be less. If the code coverage requirement isn't met, write more tests and include them in the deployment.

To run a subset of tests, set the `RunSpecifiedTests` test level on the `DeployOptions` object. Next, specify each test class to run in `DeployOptions`. Finally, pass `DeployOptions` as an argument to the `deploy()` call. The following example performs those steps to run only the specified test classes.

```
// Create the DeployOptions object.
DeployOptions deployOptions = new DeployOptions();

// Set the appropriate test level.
deployOptions.setTestLevel(TestLevel.RunSpecifiedTests);

// Specify the test classes to run.
// String array contains test class names.
String[] tests = {"TestClass1", "TestClass2", "TestClass3"};
// Add the test class names array to the deployment options.
deployOptions.setRunTests(tests);

// Call deploy() by passing the deployment options object as an argument.
AsyncResult asyncResult = metadatabinding.deploy(zipBytes,deployOptions);
```

Notes About Running Specific Tests

- You can specify only test classes. You can't specify individual test methods.
- We recommend that you refactor test classes to include the minimum number of tests that meet code coverage requirements. Refactoring your test classes can contribute to shorter test execution times, and as a result, shorter deployment times.
- You can deactivate a trigger in the target organization by deploying it with an inactive state. However, the trigger must have been previously deployed with an active state.

Run the Same Tests in Sandbox and Production Deployments

Starting in API version 34.0, you can choose which tests to run in your development environment, such as only local tests, to match the tests run in production. In earlier versions, if you enabled tests in your sandbox deployment, you couldn't exclude managed package tests.

By default, no tests are run in a deployment to a non-production organization, such as a sandbox or a Developer Edition organization. To specify tests to run in your development environment, set a `testLevel` deployment option. For example, to run local tests in a deployment and to exclude managed package tests, set `testLevel` on the `DeployOptions` object to `TestLevel.RunLocalTests`. Next, pass this object as an argument to the `deploy()` call as follows.

```
// Create the DeployOptions object.
DeployOptions deployOptions = new DeployOptions();

// Set the appropriate test level.
deployOptions.setTestLevel(TestLevel.RunLocalTests);

// Call deploy() by passing the deployment options object as an argument.
AsyncResult asyncResult = metadatabinding.deploy(zipBytes,deployOptions);
```

Note: The `RunLocalTests` test level is enforced regardless of the contents of the deployment package. In contrast, tests are executed by default in production only if your deployment package contains Apex classes or triggers. You can use `RunLocalTests` for sandbox and production deployments.

Maintaining User References

User fields are preserved during a metadata deployment.

When a component in your deployment refers to a specific user, such as a recipient of a workflow email notification or a dashboard running user, then Salesforce attempts to locate a matching user in the destination organization by comparing usernames during the deployment.

For example, when you copy data to a sandbox, the fields containing usernames from the production organization are altered to include the sandbox name. In a sandbox named `test`, the username `user@acme.com` becomes `user@acme.com.test`. When you deploy the metadata in the sandbox to another organization, the `test` in the username is ignored.

For user references in deployments, Salesforce performs the following sequence:

1. Salesforce compares usernames in the source environment to the destination environment and adapts the organization domain name.
2. If two or more usernames match, Salesforce lists the matching names and requests one of the users in the source environment be renamed.
3. If a username in the source environment doesn't exist in the destination environment, Salesforce displays an error, and the deployment stops until the usernames are removed or resolved to users in the destination environment.

CHAPTER 4 CRUD-Based Metadata Development

Use the CRUD-based metadata calls to create, update, or delete setup and configuration components for your organization or application. These configuration components include custom objects, custom fields, and other configuration metadata. The metadata calls mimic the behavior in the Salesforce user interface for creating, updating, or deleting components. Whatever rules apply there also apply to these calls.

Metadata calls are different from the core, synchronous API calls in the following ways:

- Metadata API calls are available in a separate WSDL. To download the WSDL, log into Salesforce, from Setup, enter *`API`* in the `Quick Find` box, then select **API** and click the **Download Metadata WSDL** link.
- After logging in, you must send Metadata API calls to the Metadata API endpoint, which has a different URL than the SOAP API. Retrieve the `metadataServerUrl` from the LoginResult returned by your SOAP API `login()` call. For more information about the SOAP API, see the *SOAP API Developer Guide*.
- Metadata calls are either synchronous or asynchronous. CRUD calls are synchronous in API version 30.0 and later, and similar to the API core calls the results are returned in a single call. In earlier API versions, create, update, and delete are only asynchronous, which means that the results are not immediately returned in one call.
- There are synchronous metadata calls that map to the corresponding core SOAP API synchronous calls.
  - `createMetadata()` maps to the `create()` SOAP API call.
  - `updateMetadata()` maps to the `update()` SOAP API call.
  - `deleteMetadata()` maps to the `delete()` SOAP API call.

Note: Metadata API also supports `retrieve()` and `deploy()` calls for retrieving and deploying metadata components. For more information, see Deploying and Retrieving Metadata.

Java Sample for CRUD-Based Development with Synchronous Calls

This section guides you through a sample Java client application that uses CRUD-based calls. This sample application performs the following main tasks:

1. Uses the `MetadataLoginUtil.java` class to create a Metadata connection. For more information, see Step 3: Walk Through the Java Sample Code.
2. Calls `createMetadata()` to create a custom object. This call returns the result in one call.
3. Inspects the returned `SaveResult` object to check if the operation succeeded, and if it didn't, writes the component name, error message, and status code to the output.

```
import com.sforce.soap.metadata.*;

/**
 * Sample that logs in and creates a custom object through the metadata API
 */
public class CRUDSampleCreate {
    private MetadataConnection metadataConnection;

    // one second in milliseconds
    private static final long ONE_SECOND = 1000;

    public CRUDSampleCreate() {
    }

    public static void main(String[] args) throws Exception {
        CRUDSampleCreate crudSample = new CRUDSampleCreate();
        crudSample.runCreate();
    }

    /**
     * Create a custom object. This method demonstrates usage of the
     * create() and checkStatus() calls.
     *
     * @param uniqueName Custom object name should be unique.
     */
    private void createCustomObjectSync(final String uniqueName) throws Exception {
        final String label = "My Custom Object";
        CustomObject co = new CustomObject();
        co.setFullName(uniqueName);
        co.setDeploymentStatus(DeploymentStatus.Deployed);
        co.setDescription("Created by the Metadata API Sample");
        co.setEnableActivities(true);
        co.setLabel(label);
        co.setPluralLabel(label + "s");
        co.setSharingModel(SharingModel.ReadWrite);

        // The name field appears in page layouts, related lists, and elsewhere.
        CustomField nf = new CustomField();
        nf.setType(FieldType.Text);
        nf.setDescription("The custom object identifier on page layouts, related lists
etc");
        nf.setLabel(label);
        nf.setFullName(uniqueName);
        customObject.setNameField(nf);

        SaveResult[] results = metadataConnection
                .createMetadata(new Metadata[] { co });

        for (SaveResult r : results) {
            if (r.isSuccess()) {
                System.out.println("Created component: " + r.getFullName());
            } else {
                System.out
                        .println("Errors were encountered while creating "
                                + r.getFullName());
                for (Error e : r.getErrors()) {
                    System.out.println("Error message: " + e.getMessage());
                    System.out.println("Status code: " + e.getStatusCode());
                }
            }
        }
    }

    private void runCreate() throws Exception {
        metadataConnection = MetadataLoginUtil.login();
        // Custom objects and fields must have __c suffix in the full name.
        final String uniqueObjectName = "MyCustomObject__c";
        createCustomObjectSync(uniqueObjectName);
    }
}
```

Java Sample for CRUD-Based Development with Asynchronous Calls

Important: The sample in this section depends on the asynchronous `create()` CRUD call. Asynchronous CRUD calls are no longer available as of API version 31.0 and are available only in earlier API versions.

This section guides you through a sample Java client application that uses asynchronous CRUD-based calls. This sample application performs the following main tasks:

1. Uses the `MetadataLoginUtil.java` class to create a Metadata connection. For more information, see Step 3: Walk Through the Java Sample Code.
2. Calls `create()` to create a new custom object.

   Salesforce returns an AsyncResult object for each component you tried to create. The AsyncResult object is updated with status information as the operation moves from a queue to completed or error state.

3. Calls `checkStatus()` in a loop until the status value in AsyncResult indicates that the create operation is completed.

Note the error handling code that follows each API call.

```
import com.sforce.soap.metadata.*;

/**
 * Sample that logs in and creates a custom object through the metadata api
 */
public class CRUDSample {
    private MetadataConnection metadataConnection;

    // one second in milliseconds
    private static final long ONE_SECOND = 1000;

    public CRUDSample() {
    }

    public static void main(String[] args) throws Exception {
        CRUDSample crudSample = new CRUDSample();
        crudSample.runCreate();
    }

    /**
     * Create a custom object. This method demonstrates usage of the
     * create() and checkStatus() calls.
     *
     * @param uniqueName Custom object name should be unique.
     */
    private void createCustomObject(final String uniqueName) throws Exception {
        final String label = "My Custom Object";
        CustomObject customObject = new CustomObject();
        customObject.setFullName(uniqueName);
        customObject.setDeploymentStatus(DeploymentStatus.Deployed);
        customObject.setDescription("Created by the Metadata API Sample");
        customObject.setLabel(label);
        customObject.setPluralLabel(label + "s");
        customObject.setSharingModel(SharingModel.ReadWrite);

        // The name field appears in page layouts, related lists, and elsewhere.
        CustomField nf = new CustomField();
        nf.setType(FieldType.Text);
        nf.setDescription("The custom object identifier on page layouts, related lists
etc");
        nf.setLabel(label);
        nf.setFullName(uniqueName);
        customObject.setNameField(nf);

        AsyncResult[] asyncResults = metadataConnection.create(
            new CustomObject[]{customObject});
        if (asyncResults == null) {
            System.out.println("The object was not created successfully");
            return;
        }

        long waitTimeMilliSecs = ONE_SECOND;

       // After the create() call completes, we must poll the results of the checkStatus()

        // call until it indicates that the create operation has completed.
        do {
            printAsyncResultStatus(asyncResults);
            waitTimeMilliSecs *= 2;
            Thread.sleep(waitTimeMilliSecs);
            asyncResults = metadataConnection.checkStatus(new
String[]{asyncResults[0].getId()});
        } while (!asyncResults[0].isDone());

        printAsyncResultStatus(asyncResults);
    }

    private void printAsyncResultStatus(AsyncResult[] asyncResults) throws Exception {
        if (asyncResults == null || asyncResults.length == 0 || asyncResults[0] == null)
{
            throw new Exception("The object status cannot be retrieved");
        }

       AsyncResult asyncResult = asyncResults[0]; //we are creating only 1 metadata object


        if (asyncResult.getStatusCode() != null) {
            System.out.println("Error status code: " +
                    asyncResult.getStatusCode());
            System.out.println("Error message: " + asyncResult.getMessage());
        }

        System.out.println("Object with id:" + asyncResult.getId() + " is " +
            asyncResult.getState());
    }

    private void runCreate() throws Exception {
        metadataConnection = MetadataLoginUtil.login();
        // Custom objects and fields must have __c suffix in the full name.
        final String uniqueObjectName = "MyCustomObject__c";
        createCustomObject(uniqueObjectName);
    }
}
```

CHAPTER 5 REST Resources

In this chapter ...

Use the REST resource `deployRequest` to move metadata (XML files) between a Salesforce organization and a local file system.

Data in XML files is formatted using the English (United States) locale. This approach ensures that fields that depend on locale, such as date fields, are interpreted consistently during data migrations between organizations using different languages. Organizations can support multiple languages for presentation to their users.

Metadata deployment is used primarily for the following development scenarios.

- Development of a custom application (or customization) in a sandbox organization. After development and testing are completed, the application or customization is then deployed into a production organization using Metadata API.
- Team development of an application in a Developer Edition organization. After development and testing are completed, you can then distribute the application via Lightning Platform AppExchange.

Working with the Zip File

The `deployRequest` resource is used to deploy a .zip file. Within the .zip file is a project manifest (`package.xml`) that lists what to retrieve or deploy, and one or more XML components that are organized into folders.

Note: A component is an instance of a metadata type. For example, `CustomObject` is a metadata type for custom objects, and the `MyCustomObject__c` component is an instance of a custom object.

The files that are deployed in a .zip file might be unpackaged components that reside in your organization (such as *standard objects*). The files might also be packaged components that reside within named packages.

Note: You can deploy up to 10,000 files at once. (In API version 43.0 and later, AppExchange packages can contain up to 12,500 files.) The .zip file size limit that applies to SOAP calls doesn't apply to the `deployRequest` REST resource. However, the 400-MB limit for components that are uncompressed into an unzipped folder after upload applies to both SOAP and REST deployments.

Every .zip file contains a project manifest, a file that's named `package.xml`, and a set of directories that contain the components. The manifest file defines the components that you're trying to retrieve or deploy and the API version used for the deployment or retrieval.

The following is a sample `package.xml` file.

```
<?xml version="1.0" encoding="UTF-8"?>
<Package xmlns="http://soap.sforce.com/2006/04/metadata">
```

```
    <types>
        <members>MyCustomObject__c</members>
        <name>CustomObject</name>
    </types>
    <types>
        <members>*</members>
        <name>CustomTab</name>
    </types>
    <types>
        <members>Standard</members>
        <name>Profile</name>
    </types>
    <version>48.0</version>
</Package>
```

The following elements can be defined in `package.xml`.

- `<fullName>` contains the name of the server-side package. If no `<fullName>` exists, this is a client-side `unpackaged` package.
- `<types>` contains the name of the metadata type (for example, `CustomObject`) and the named members (for example, `myCustomObject__c`) to be deployed. You can add multiple `<types>` elements in a manifest file.
- `<members>` contains the `fullName` of the component, such as `MyCustomObject__c`. For many metadata types, you can replace the value in `members` with the wildcard character `*` (asterisk) instead of listing each member separately. For a list of metadata types that allow the wildcard character, see the "Allows Wildcard (*)?" column in Metadata Types.

  Note: You specify Security in the `<members>` element and Settings in the name element when retrieving the SecuritySettings component type.

- `<name>` contains the metadata type, for example `CustomObject` or `Profile`. There is one name defined for each metadata type in the directory. Any metadata type that extends Metadata is a valid value. The name that's entered must match a metadata type that's defined in the Metadata API WSDL. See Metadata Types for a list.
- `<version>` is the API version number that's used when the .zip file is deployed or retrieved. Currently the valid value is `48.0`.

For more sample `package.xml` manifest files that show you how to work with different subsets of metadata, see Sample `package.xml` Manifest Files.

To delete components, see Deleting Components from an Organization.

Deploy Metadata with Apex Testing Using REST

Deploy using the `deployRequest` REST resource to initiate a request that handles all operations for the deployment.

You can deploy or retrieve up to 10,000 files at once. AppExchange packages use different limits. In API version 43.0 and 44.0, AppExchange packages can contain up to 12,500 files. In API version 45.0, AppExchange packages can contain up to 17,500 files. In API version 46.0, AppExchange packages can contain up to 22,000 files. In API version 47.0 and later, AppExchange packages can contain up to 30,000 files. The maximum size of the deployed or retrieved .zip file is 39 MB. If the files are uncompressed in an unzipped folder, the size limit is 400 MB.

URI

```
https://host/services/data/vXX.0/metadata/deployRequest
```

Formats

JSON

HTTP Method

POST

Authentication

```
Authorization: Bearer token
```

deployOptions Parameters

Note: To review the default testing behavior for deployments and approaches that can save time while still enabling you to meet testing requirements, see Running Tests in a Deployment and Run the Same Tests in Sandbox and Production Deployments.

Parameter	Description
`allowMissingFiles`	Boolean. If files that are specified in `package.xml` are not in the `.zip` file, specifies whether a deployment can still succeed. Do not set this argument for deployment to production orgs.
`autoUpdatePackage`	Reserved for future use.
`checkOnly`	Boolean. Defaults to `false`. Set to `true` to perform a test deployment (validation) of components without saving the components in the target org. A validation enables you to verify the results of tests that would be generated in a deployment, but doesn’t commit any changes. After a validation finishes with passing tests, it might qualify for deployment without rerunning tests. See Deploy a Recently Validated Component Set Without Tests.
`ignoreWarnings`	Boolean. Indicates whether a deployment is allowed to complete successfully despite one or more warnings (`true`) or not (`false`). Defaults to `false`. The DeployMessage object for a warning contains the following values: • `problemType`—`Warning` • `problem`—The text of the warning. If a warning occurs and `ignoreWarnings` is set to `true`, the `success` field in DeployMessage is `true`. If `ignoreWarnings` is set to `false`, `success` is set to `false` and the warning is treated like an error.
`performRetrieve`	Reserved for future use.

Parameter	Description
purgeOnDelete	Boolean. If true, the deleted components in the destructiveChanges.xml manifest file aren't stored in the Recycle Bin. Instead, they become immediately eligible for deletion. This option only works in Developer Edition or sandbox orgs. It doesn't work in production orgs.
rollbackOnError	Boolean. Indicates whether any failure causes a complete rollback (true) or not (false). If false, whatever actions can be performed without errors are performed, and errors are returned for the remaining actions. This parameter must be set to true if you are deploying to a production org. The default is false.
runTests	String[]. A list of Apex tests to run during deployment. Specify the class name, one name per instance. The class name can also specify a namespace with a dot notation. For more information, see Running a Subset of Tests in a Deployment. To use this option, set testLevel to RunSpecifiedTests.
singlePackage	Boolean. Indicates whether the specified .zip file points to a directory structure with a single package (true) or a set of packages (false).
testLevel	TestLevel (enumeration of type string). Optional. Specifies which tests are run as part of a deployment. The test level is enforced regardless of the types of components that are present in the deployment package. Valid values are: - NoTestRun—No tests are run. This test level applies only to deployments to development environments, such as sandbox, Developer Edition, or trial organizations. This test level is the default for development environments. - RunSpecifiedTests—Only the tests that you specify in the runTests option are run. Code coverage requirements differ from the default coverage requirements when using this test level. Each class and trigger in the deployment package must be covered by the executed tests for a minimum of 75% code coverage. This coverage is computed for each class and triggers individually and is different than the overall coverage percentage. - RunLocalTests—All tests in your org are run, except the ones that originate from installed managed and unlocked packages. This test level is the default for production deployments that include Apex classes or triggers. - RunAllTestsInOrg—All tests are run. The tests include all tests in your org, including tests of managed packages. If you don’t specify a test level, the default test execution behavior is used. See Running Tests in a Deployment. Note: Apex tests that run as part of a deployment always run synchronously and serially.

Request Body: Deploy Metadata

When you deploy metadata, your request includes both the deployment parameters and the .zip file containing the component directories and the manifest. Set the header to `Content-Type: multipart/form-data`,

This example POST request creates a `deployRequest` object that initiates a deployment.

1. The POST request header is set to `Content-Type: multipart/form-data` and defines a `boundary` value to encapsulate different subparts of the request.
2. In the subpart after the first boundary, a JSON request creates a `deployOptions` child object for passing the deployment parameters.
3. The subpart after the second boundary specifies the .zip file containing the manifest and the component directories.

```
POST /services/data/v48.0/metadata/deployRequest
Authorization: Bearer 00D....
Content-Type: multipart/form-data; boundary=--------------------------BOUNDARY
----------------------------BOUNDARY
Content-Disposition: form-data; name="json"
Content-Type: application/json
{
    "deployOptions" :
        {
        "allowMissingFiles" : false,
        "autoUpdatePackage" : false,
        "checkOnly" : false,
        "ignoreWarnings" : false,
        "performRetrieve" : false,
        "purgeOnDelete" : false,
        "rollbackOnError" : false,
        "runTests" : null,
        "singlePackage" : true,
        "testLevel" : "RunAllTestsInOrg"
        }
    }
----------------------------BOUNDARY
Content-Disposition: form-data; name="file"; filename="deploy.zip"
Content-Type: application/zip

//Contents of deploy.zip
----------------------------BOUNDARY--
```

Response Body: Deploy Metadata

When an HTTP status code of `201` (Created) is returned, your request has succeeded and resulted in the creation of a deployment that is being processed.

```
{ "id" : "0Afxx00000001VPCAY",
  "deployOptions" :
   { "checkOnly" : false,
     "singlePackage" : false,
     "allowMissingFiles" : false,
     "performRetrieve" : false,
     "autoUpdatePackage" : false,
     "rollbackOnError" : true,
     "ignoreWarnings" : false,
     "purgeOnDelete" : false,
     "runAllTests" : false },
  "deployResult" :
   { "id" : "0Afxx00000001VPCAY",
     "success" : false,
     "checkOnly" : false,
     "ignoreWarnings" : false,
     "rollbackOnError" : true,
     "status" : "Pending",
     "runTestsEnabled" : false,
     "done" : false } }
```

deployResult Parameters

Parameter	Description
`id`	ID. ID of the component being deployed.
`canceledBy`	ID. The ID of the user who canceled the deployment.
`canceledByName`	String. The full name of the user who canceled the deployment.
`checkOnly`	Boolean. Indicates whether this deployment is being used to check the validity of the deployed files without making any changes in the org (`true`) or not (`false`). A check-only deployment doesn't deploy any components or change the organization in any way.
`completedDate`	DateTime. Timestamp for when the deployment process ended.
`createdBy`	ID. The ID of the user who created the deployment.
`createdByName`	String. The full name of the user who created the deployment.
`createdDate`	DateTime. Timestamp for when the deploy request was received.
`details`	DeployDetails. Provides the details of a deployment that is in-progress or ended if `?includeDetails=true` is added as a query to the GET request.
`done`	Boolean. Indicates whether the server finished processing the deploy request for the specified `id`.
`errorMessage`	String. Message corresponding to the values in the `errorStatusCode` field, if any.
`errorStatusCode`	String. If an error occurred during the deploy request, a status code is returned, and the message corresponding to the status code is returned in `errorMessagefield`.
`ignoreWarnings`	Boolean. Optional. Defaults to `false`. Specifies whether a deployment continues even if the deployment generates warnings. Do not set this argument to `true` for deployments to production organizations.
`lastModifiedDate`	DateTime. Timestamp of the last update for the deployment process.

Parameter	Description
`numberComponentErrors`	Int. The number of components deployed in the deployment process. Use this value with the `numberComponentsTotal` value to get an estimate of the deployment's progress.
`numberComponentsTotal`	Int. The total number of components in the deployment. Use this value with the `numberComponentsDeployed` value to get an estimate of the deployment's progress.
`numberTestErrors`	Int. The number of Apex tests that have generated errors during this deployment.
`numberTestsCompleted`	The number of completed Apex tests for this deployment. Use this value with the `numberTestsTotal` value to get an estimate of the deployment's test progress.
`numberTestsTotal`	Int. The total number of Apex tests for this deployment. Use this value with the `numberTestsCompleted` value to get an estimate of the deployment's test progress. The value in this field is not accurate until the deployment has started running tests for the components being deployed.
`runTestsEnabled`	Boolean. Indicates whether Apex tests were run as part of this deployment (`true`) or not (`false`). Tests are either automatically run as part of a deployment or can be set to run in the deployOptions child object.
`rollbackOnError`	Boolean. Defaults to `true`. Indicates whether any failure causes a complete rollback (`true`) or not (`false`). If `false`, whatever set of actions can be performed without errors are performed, and errors are returned for the remaining actions. This parameter must be set to `true` if you are deploying to a production org.
`startDate`	DateTime. Timestamp for when the deployment process began.
`stateDetail`	String. Indicates which component is being deployed or which Apex test class is running.
`status`	Indicates the current state of the deployment. The valid values are: • `Pending` • `InProgress` • `Succeeded` • `SucceededPartial` • `Failed` • `Canceling` • `Canceled`
`success`	Boolean. Indicates whether the deployment was successful (`true`) or not (`false`).

Check the Status of Your Deployment Using REST Resources

Check the status of your deployment by using passing the deployment request ID in the URL The response body is similar to that returned by the original deployment request, but it includes information about the deployment in progress.

URI

https://***host***/services/data/v***XX***.0/metadata/deployRequest/***deployRequestId***

To include more details in the response, use:

`https://`***`host`***`/services/data/v`***`XX`***`.0/metadata/deployRequest/`***`deployRequestId`***`?includeDetails=true`

Formats

JSON

HTTP Method

GET

Authentication

`Authorization: Bearer` ***`token`***

Response Body: Deploy Metadata

The following example shows the response when `?includeDetails=true` is added as a query to the GET request.

```
    {
        "id" : "0Afxx00000000lWCAQ"
        "url" :
"https://host/services/data/vXX.0/metadata/deployRequest/0Afxx00000000lWCAQ?includeDetails=true",

        "deployResult" :
            {
            "checkOnly" : "false",
            "ignoreWarnings" : "false",
            "rollbackOnError" : "false",

            "status : "InProgress",
            "numberComponentsDeployed" : "10",
            "numberComponentsTotal" : "1032",
            "numberComponentErrors" : "0",
            "numberTestsCompleted" : "45",
            "numberTestsTotal" : "135",
            "numberTestErrors" : "0",
            "details" :  {
              "componentFailures" : [],
              "componentSuccesses" : [],
                    "retrieveResult" : null,
                    "runTestResults" : {
                    "numRun" : 0,
                    "successes" : [ … ],
                    "failures" : []
                    }
            },

            "createdDate" : "2017-10-10T08:22Z",
            "startDate" : "2017-10-10T08:22Z",
            "lastModifiedDate" : "2017-10-10T08:44Z",
            "completedDate" : "2017-10-10T08:44Z",

            "errorStatusCode" : null,
            "errorMessage" : null,
            "stateDetail" : "Processing Type: Apex Component",
            "createdBy" : "005xx0000001Sv1m",
            "createdByName" : "stephanie stevens",
            "canceledBy" : null,
            "canceledByName" : null,
            "isRunTestsEnabled" : null
            }

       "deployOptions": {
             "allowMissingFiles" : false,
             "autoUpdatePackage" : false,
             "checkOnly" : true,
             "ignoreWarnings" : false,
             "performRetrieve" : false,
             "purgeOnDelete" : false,
             "rollbackOnError" : false,
             "runTests" : null,
             "singlePackage" : true,
             "testLevel" : "RunAllTestsInOrg"
             }
     }
```

Expect an HTTP status code of `200` (OK) to be returned.

Deploy a Recently Validated Component Set Without Tests

You can deploy components to production in less time by skipping the execution of Apex tests when testing requirements have already been met.

- The components have been validated successfully for the target environment within the last 10 days.
- As part of the validation, Apex tests in the target org have passed.
- Code coverage requirements are met.
  - If all tests in the org or all local tests are run, overall code coverage is at least 75%, and Apex triggers have some coverage.
  - If specific tests are run with the `RunSpecifiedTests` test level, each class and trigger to be deployed is covered by at least 75% individually.

This operation is equivalent to performing a quick deployment of a recent validation on the Deployment Status page in the Salesforce user interface.

To validate but not deploy a set of components when using the `deployRequest` resource, set the `checkOnly` parameter of `deployOptions` to `true`. Note the deployment request ID in the response. Use this ID (associated with a successful validation) later to deploy the component set without repeating the validation.

URI

`https://`***`host`***`/services/data/v`***`XX.`***`0/metadata/deployRequest/`***`validatedDeployRequestId`***

Formats

JSON

HTTP Method

POST

Authentication

`Authorization: Bearer` ***`token`***

Request Body: Deploy a Recently Validated Component Set Without Tests

Note: The HTTP method for deploying a recently validated component set is POST, not PATCH. Using PATCH would create a new deployment.

```
    {
       "validatedDeployRequestId" : "0Afxx00000000lWCAQ"
    }
```

If there is no corresponding deployment package that meets the validation requirements, you receive an HTTP status code of `404` (Not Found). If the validated deployment package is found, the HTTP status code returned is `201` (Created).

Response Body: Deploy a Recently Validated Component Set Without Tests

Note: The response body from the deployment without validation request includes a new request ID, because it is separate from the earlier request for a validation-only deployment.

```
    {
       "validatedDeployRequestId" : "0Afxx00000000lWCAQ"
       "id" : "0Afxx00000000lWMEM"
      "url" : "https://host/services/data/vXX.0/metadata/deployRequest/0Afxx00000000lWMEM",

       "deployOptions" :
             {
             "allowMissingFiles" : false,
             "autoUpdatePackage" : false,
             "checkOnly" : true,
             "ignoreWarnings" : false,
             "performRetrieve" : false,
             "purgeOnDelete" : false,
             "rollbackOnError" : false,
             "runTests" : null,
             "singlePackage" : true,
             "testLevel" : "RunAllTestsInOrg"
}
     }
```

When an HTTP status code of `201` (Created) is returned, your request has succeeded and resulted in the creation of a deployment that is being processed. In the preceding example response body, the ID of the validation-only deployment request is `0Afxx00000000lWCAQ`; the ID of the deployment without validation request is `0Afxx00000000lWMEM`.

Cancel a Deployment in Progress Using REST

You can request a cancellation of a deployment that's already in progress. Make the cancellation request by patching the status of an ongoing `deployRequest`. The cancellation is processed asynchronously.

URI

`https://`***`host`***`/services/data/v`***`XX`***`.0/metadata/deployRequest/`***`deployRequestId`***

Formats

JSON

HTTP Method

PATCH

Authentication

`Authorization: Bearer` ***`token`***

Request Body: Request Deployment Cancelation

The JSON request body for a deployment cancellation includes a PATCH to the status of the original `deployRequest`.

```
{
 "deployResult":
       {
       "status" : "Canceling"
       }
}
```

Response Body: Request Deployment Cancelation

Because the cancellation request is processed asynchronously, the status shown in the response body can be either `Canceling` or `Canceled`.

```
{
   "id" : "0Afxx00000000lWCAQ"
  "url" : “https://host/services/data/vXX.0/metadata/deployRequest/0Afxx00000000lWCAQ",

   "deployResult":
         {
         "checkOnly" : "false",
         "ignoreWarnings" : "false",
         "rollbackOnError" : "false",
         "status : "Canceling",  // or Canceled
         "numberComponentsDeployed" : "10",
         "numberComponentsTotal" : "1032",
         "numberComponentErrors" : "0",
         "numberTestsCompleted" : "45",
         "numberTestsTotal" : "135",
         "numberTestErrors" : "0",
         "details" :  {
            "componentFailures" : [],
            "componentSuccesses" : [],
                  "retrieveResult" : null,
                  "runTestResults” : {
                     "numRun" : 0,
                     "successes" : [ … ],
                     "failures" : []
                  }
            },

            "createdDate" : "2017-10-10T08:22Z",
            "startDate" : "2017-10-10T08:22Z",
            "lastModifiedDate" : "2017-10-10T08:44Z",
            "completedDate" : "2017-10-10T08:44Z",
            "errorStatusCode" : null,
            "errorMessage" : null,
            "stateDetail" : "Processing Type: Apex Component",
            "createdBy" : "005xx0000001Sv1m",
            "createdByName" : "steve stevens",
            "canceledBy" : null,
            "canceledByName" : null,
            "isRunTestsEnabled" : null
            }
   }
```

When an HTTP status code of `202` (Accepted) is returned, your cancelation request is in progress or successful.

Use the Metadata REST API with the Salesforce CLI

You can use the Metadata REST API with all deployments, either via the `deployRequest` REST resource or via the Salesforce CLI. You can enable the Salesforce CLI to use Metadata REST API deployments by using a CLI runtime configuration value.

You must have the Modify Metadata Through Metadata API Functions permission or the Modify All Data permission to perform deployments. This requirement includes deployments using Metadata REST API for Deploy.

By default, the Salesforce CLI uses SOAP for deployments. Deployments using REST aren't bound by the 39 MB .zip file size limit that applies to SOAP deployments. The limit for the number of components that are uncompressed into an unzipped folder after upload is the same for both SOAP and REST deployments.

You enable the Metadata REST API for Deploy in the Salesforce CLI by using a CLI runtime configuration value. For more about setting CLI runtime configuration values, see the *Salesforce DX Developer Guide*.

- Set the Salesforce CLI to use the Metadata REST API for all deployments in the current project:

  ```
  sfdx force:config:set restDeploy=true
  ```

  After Metadata REST API for Deploy is enabled for the Salesforce CLI, deployments initiated using the `mdapi:deploy` or `source:push` commands use REST.

CHAPTER 6 Error Handling

Metadata API calls return error information that your client application can use to identify and resolve runtime errors. The Metadata API provides the following types of error handling:

- Since the Metadata API uses the enterprise or partner WSDLs to authenticate, it uses SOAP fault messages defined in those WSDLs for errors resulting from badly formed messages, failed authentication, or similar problems. Each SOAP fault has an associated ExceptionCode. For more details, see “Error Handling” in the *SOAP API Developer Guide*.
- For errors with the asynchronous `create()`, `update()`, and `delete()` calls, see the error status code in the `statusCode` field in the AsyncResult object for the associated component.
- For errors with the synchronous CRUD calls, see the error status code in the `statusCode` field of the Error object corresponding to each error in the array returned by the `errors` field of the appropriate result object. For example, the result object of `createMetadata()` is SaveResult.
- For errors with `deploy()`, see the `problem` and `success` fields in the DeployMessage object for the associated component.
- For errors with `retrieve()`, see the `problem` field in the RetrieveMessage object for the associated component.

For sample code, see Step 3: Walk Through the Java Sample Code on page 7.

Error Handling for Session Expiration

When you sign on via the `login()` call, a new client session begins and a corresponding unique session ID is generated. Sessions automatically expire after the amount of time specified in the **Security Controls** setup area of the Salesforce application (default two hours). When your session expires, the exception code INVALID_SESSION_ID is returned. If this happens, you must invoke the `login()` call again. For more information about `login()`, see the *SOAP API Developer Guide*.

REFERENCE

CHAPTER 7 File-Based Calls

Use file-based calls to deploy or retrieve XML components.

- deploy()
- deployRecentValidation()
- retrieve()

deploy()

Uses file representations of components to create, update, or delete those components in a Salesforce org.

Syntax

```
AsyncResult = metadatabinding.deploy(base64 zipFile, DeployOptions deployOptions)
```

Usage

Use this call to take file representations of components and deploy them into an org by creating, updating, or deleting the components they represent.

Note: You can deploy or retrieve up to 10,000 files at once. AppExchange packages use different limits. In API version 43.0 and 44.0, AppExchange packages can contain up to 12,500 files. In API version 45.0, AppExchange packages can contain up to 17,500 files. In API version 46.0, AppExchange packages can contain up to 22,000 files. In API version 47.0 and later, AppExchange packages can contain up to 30,000 files. The maximum size of the deployed or retrieved .zip file is 39 MB. If the files are uncompressed in an unzipped folder, the size limit is 400 MB.

- If using the Ant Migration Tool to deploy an unzipped folder, all files in the folder are compressed first. The maximum size of uncompressed components in an unzipped folder is 400 MB or less depending on the compression ratio. If the files have a high compression ratio, you can migrate a total of approximately 400 MB because the compressed size would be under 39 MB. However, if the components can't be compressed much, like binary static resources, you can migrate less than 400 MB.
- Metadata API base-64 encodes components after they're compressed. The resulting .zip file can't exceed 50 MB, which is the limit for SOAP messages. Base-64 encoding increases the size of the payload, so your compressed payload can't exceed approximately 39 MB before encoding.
- You can perform a `retrieve()` call for a big object only if its index is defined. If a big object is created in Setup and doesn't yet have an index defined, you can't retrieve it.

In API version 29.0, Salesforce improved the deployment status properties and removed the requirement to use `checkStatus()` after a `deploy()` call to get information about deployments. Salesforce continues to support the use of `checkStatus()` when using `deploy()` with API version 28.0 or earlier.

For API version 29.0 or later, deploy (create or update) packaged or unpackaged components using the following steps.

1. Issue a `deploy()` call to start the asynchronous deployment. An AsyncResult object is returned. Note the value in the `id` field and use it for the next step.
2. Issue a `checkDeployStatus()` call in a loop until the `done` field of the returned DeployResult contains `true`, which means that the call is completed. The DeployResult object contains information about an in-progress or completed deployment started using the `deploy()` call. When calling `checkDeployStatus()`, pass in the `id` value from the AsyncResult object from the first step.

For API version 28.0 or earlier, deploy (create or update) packaged or unpackaged components using the following steps.

1. Issue a `deploy()` call to start the asynchronous deployment. An AsyncResult object is returned. If the call is completed, the `done` field contains `true`. Most often, the call is not completed quickly enough to be noted in the first result. If it is completed, note the value in the `id` field returned and skip the next step.
2. If the call is not complete, issue a `checkStatus()` call in a loop. In the loop, use the value in the `id` field of the AsyncResult object returned by the `deploy()` call in the previous step. Check the AsyncResult object which is returned until the `done` field contains `true`. The time taken to complete a `deploy()` call depends on the size of the zip file being deployed. Therefore, use a longer wait time between iterations as the size of the zip file increases.
3. Issue a `checkDeployStatus()` call to obtain the results of the `deploy()` call, using the `id` value returned in the first step.

To track the status of deployments that are in progress or completed in the last 30 days, from Setup, enter *`Deployment Status`* in the Quick Find box, then select **Deployment Status**.

You can cancel a deployment while it's in progress or in the queue by clicking **Cancel** next to the deployment. The deployment then has the status `Cancel Requested` until the deployment is completely canceled. A canceled deployment is listed in the Failed section.

The `package.xml` file is a project manifest that lists all the components that you want to retrieve or deploy. You can use `package.xml` to add components. To delete components, add another manifest file. See Deleting Components from an Organization.

Permissions

Your client application must be logged in with the Modify All Data or Modify Metadata Through Metadata API Functions permission.

Note: If a user requires access to metadata but not to data, you can enable the Modify Metadata Through Metadata API Functions permission to give the access the user needs without providing access to org data.

Arguments

Name	Type	Description
`zipFile`	base64	Base 64-encoded binary data. Client applications must encode the binary data as base64.
`deployOptions`	DeployOptions	Encapsulates options for determining which packages or files are deployed.

DeployOptions

The following deployment options can be selected for this call:

Name	Type	Description
`allowMissingFiles`	boolean	If files that are specified in `package.xml` are not in the `.zip` file, specifies whether a deployment can still succeed. Do not set this argument for deployment to *production orgs*.
`autoUpdatePackage`	boolean	If a file is in the `.zip` file but not specified in `package.xml`, specifies whether the file is automatically added to the package. A `retrieve()` is issued with the updated `package.xml` that includes the `.zip` file. Do not set this argument for deployment to *production orgs*.
`checkOnly`	boolean	Defaults to `false`. Set to `true` to perform a test deployment (validation) of components without saving the components in the target org. A validation enables you to verify the results of tests that would be generated in a deployment, but doesn't commit any changes. After a validation finishes with passing tests, it might qualify for deployment without rerunning tests. See deployRecentValidation(). Note: If you change a field type from Master-Detail to Lookup or vice versa, the change isn't supported when using the `checkOnly` option to test a deployment. This change isn't supported for test deployments to avoid permanently altering your data. If a change that isn't supported for test deployments is included in a deployment package, the test deployment fails and issues an error. If your deployment package changes a field type from Master-Detail to Lookup or vice versa, you can still validate the changes before you deploy to production. Perform a full deployment to another test sandbox. A full deployment includes a validation of the changes as part of the deployment process. A Metadata API deployment that includes Master-Detail relationships deletes all detail records in the Recycle Bin in the following cases. **1.** For a deployment with a new Master-Detail field, soft delete (send to the Recycle Bin) all detail records before proceeding to deploy the Master-Detail field, or the deployment fails. During the deployment, detail records are permanently deleted from the Recycle Bin and cannot be recovered. **2.** For a deployment that converts a Lookup field relationship to a Master-Detail relationship, detail records must reference a master record or be soft-deleted (sent to the Recycle Bin) for the deployment to succeed. However, a successful

Name	Type	Description
		deployment permanently deletes any detail records in the Recycle Bin.
ignoreWarnings	boolean	Indicates whether a warning should allow a deployment to complete successfully (true) or not (false). Defaults to false. The DeployMessage object for a warning contains the following values: • problemType—Warning • problem—The text of the warning. If a warning occurs and ignoreWarnings is set to true, the success field in DeployMessage is true. If ignoreWarnings is set to false, success is set to false and the warning is treated like an error. This field is available in API version 18.0 and later. Prior to version 18.0, there was no distinction between warnings and errors. All problems were treated as errors and prevented a successful deployment.
performRetrieve	boolean	Indicates whether a retrieve() call is performed immediately after the deployment (true) or not (false). Set to true to retrieve whatever was just deployed.
purgeOnDelete	boolean	If true, the deleted components in the destructiveChanges.xml manifest file aren't stored in the Recycle Bin. Instead, they become immediately eligible for deletion. This field is available in API version 22.0 and later. This option only works in Developer Edition or sandbox orgs. It doesn’t work in production orgs. Note: When you delete a roll-up summary field using Metadata API, the field isn't saved in the Recycle Bin. The field is purged even if you don’t set the purgeOnDelete deployment option to true.
rollbackOnError	boolean	Indicates whether any failure causes a complete rollback (true) or not (false). If false, whatever actions can be performed without errors are performed, and errors are returned for the remaining actions. This parameter must be set to true if you are deploying to a production org. The default is false.
runAllTests	boolean	(Deprecated and only available in API version 33.0 and earlier.) This field defaults to false. Set to true to run all Apex tests

Name	Type	Description
		after deployment, including tests that originate from installed managed packages. Note: Apex tests that run as part of a deployment always run synchronously and serially.
`runTests`	string[]	A list of Apex tests to run during deployment. Specify the class name, one name per instance. The class name can also specify a namespace with a dot notation. For more information, see Running a Subset of Tests in a Deployment. To use this option, set `testLevel` to `RunSpecifiedTests`.
`singlePackage`	boolean	Indicates whether the specified `.zip` file points to a directory structure with a single package (`true`) or a set of packages (`false`).
`testLevel`	TestLevel (enumeration of type string)	Optional. Specifies which tests are run as part of a deployment. The test level is enforced regardless of the types of components that are present in the deployment package. Valid values are: • `NoTestRun`—No tests are run. This test level applies only to deployments to development environments, such as sandbox, Developer Edition, or trial organizations. This test level is the default for development environments. • `RunSpecifiedTests`—Only the tests that you specify in the `runTests` option are run. Code coverage requirements differ from the default coverage requirements when using this test level. Each class and trigger in the deployment package must be covered by the executed tests for a minimum of 75% code coverage. This coverage is computed for each class and triggers individually and is different than the overall coverage percentage. • `RunLocalTests`—All tests in your org are run, except the ones that originate from installed managed and unlocked packages. This test level is the default for production deployments that include Apex classes or triggers. • `RunAllTestsInOrg`—All tests are run. The tests include all tests in your org, including tests of managed packages. If you don't specify a test level, the default test execution behavior is used. See Running Tests in a Deployment. Note: Apex tests that run as part of a deployment always run synchronously and serially. This field is available in API version 34.0 and later.

Response

AsyncResult

Sample Code—Java

This sample shows how to deploy components in a zip file. See the `retrieve()` sample code for details on how to retrieve a zip file.

```
package com.doc.samples;

import java.io.*;

import java.rmi.RemoteException;

import com.sforce.soap.metadata.AsyncResult;
import com.sforce.soap.metadata.DeployDetails;
import com.sforce.soap.metadata.MetadataConnection;
import com.sforce.soap.metadata.DeployOptions;
import com.sforce.soap.metadata.DeployResult;
import com.sforce.soap.metadata.DeployMessage;
import com.sforce.soap.metadata.RunTestsResult;
import com.sforce.soap.metadata.RunTestFailure;
import com.sforce.soap.metadata.CodeCoverageWarning;
import com.sforce.soap.enterprise.LoginResult;
import com.sforce.soap.enterprise.EnterpriseConnection;
import com.sforce.ws.ConnectionException;
import com.sforce.ws.ConnectorConfig;

/**
 * Deploy a zip file of metadata components.
 * Prerequisite: Have a deploy.zip file that includes a package.xml manifest file that
 * details the contents of the zip file.
 */
public class DeploySample {
    // binding for the metadata WSDL used for making metadata API calls
    private MetadataConnection metadataConnection;

    static BufferedReader rdr = new BufferedReader(new InputStreamReader(System.in));

    private static final String ZIP_FILE = "deploy.zip";

    // one second in milliseconds
    private static final long ONE_SECOND = 1000;
    // maximum number of attempts to deploy the zip file
    private static final int MAX_NUM_POLL_REQUESTS = 50;

    public static void main(String[] args) throws Exception {
        final String USERNAME = "user@company.com";
        // This is only a sample. Hard coding passwords in source files is a bad practice.

        final String PASSWORD = "password";
        final String URL = "https://login.salesforce.com/services/Soap/c/29.0";

        DeploySample sample = new DeploySample(USERNAME, PASSWORD, URL);
        sample.deployZip();
    }

    public DeploySample(String username, String password, String loginUrl)
            throws ConnectionException {
        createMetadataConnection(username, password, loginUrl);
    }

    public void deployZip()
        throws RemoteException, Exception
    {
        byte zipBytes[] = readZipFile();
        DeployOptions deployOptions = new DeployOptions();
        deployOptions.setPerformRetrieve(false);
        deployOptions.setRollbackOnError(true);
        AsyncResult asyncResult = metadataConnection.deploy(zipBytes, deployOptions);
        String asyncResultId = asyncResult.getId();

        // Wait for the deploy to complete
        int poll = 0;
        long waitTimeMilliSecs = ONE_SECOND;
        DeployResult deployResult = null;
        boolean fetchDetails;
        do {
            Thread.sleep(waitTimeMilliSecs);
            // double the wait time for the next iteration
            waitTimeMilliSecs *= 2;
            if (poll++ > MAX_NUM_POLL_REQUESTS) {
                throw new Exception("Request timed out. If this is a large set " +
                        "of metadata components, check that the time allowed by " +
                        "MAX_NUM_POLL_REQUESTS is sufficient.");
            }

            // Fetch in-progress details once for every 3 polls
            fetchDetails = (poll % 3 == 0);
          deployResult = metadataConnection.checkDeployStatus(asyncResultId, fetchDetails);

            System.out.println("Status is: " + deployResult.getStatus());
            if (!deployResult.isDone() && fetchDetails) {
                printErrors(deployResult, "Failures for deployment in progress:\n");
            }
        }
        while (!deployResult.isDone());

        if (!deployResult.isSuccess() && deployResult.getErrorStatusCode() != null) {
            throw new Exception(deployResult.getErrorStatusCode() + " msg: " +
                    deployResult.getErrorMessage());
        }

        if (!fetchDetails) {
            // Get the final result with details if we didn't do it in the last attempt.
            deployResult = metadataConnection.checkDeployStatus(asyncResultId, true);
        }
        if (!deployResult.isSuccess()) {
            printErrors(deployResult, "Final list of failures:\n");
            throw new Exception("The files were not successfully deployed");
        }

        System.out.println("The file " + ZIP_FILE + " was successfully deployed");
    }

    /**
     * Read the zip file contents into a byte array.
     * @return byte[]
     * @throws Exception - if cannot find the zip file to deploy
     */
    private byte[] readZipFile()
        throws Exception
    {
        // We assume here that you have a deploy.zip file.
        // See the retrieve sample for how to retrieve a zip file.
        File deployZip = new File(ZIP_FILE);
        if (!deployZip.exists() || !deployZip.isFile())
            throw new Exception("Cannot find the zip file to deploy. Looking for " +
                    deployZip.getAbsolutePath());

        FileInputStream fos = new FileInputStream(deployZip);
        ByteArrayOutputStream bos = new ByteArrayOutputStream();
        int readbyte = -1;
        while ((readbyte = fos.read()) != -1)  {
            bos.write(readbyte);
        }
        fos.close();
        bos.close();
        return bos.toByteArray();
    }


    /**
     * Print out any errors, if any, related to the deploy.
     * @param result - DeployResult
     */
    private void printErrors(DeployResult result, String messageHeader)
    {
        DeployDetails deployDetails = result.getDetails();

        StringBuilder errorMessageBuilder = new StringBuilder();
        if (deployDetails != null) {
            DeployMessage[] componentFailures = deployDetails.getComponentFailures();
            for (DeployMessage message : componentFailures) {
                String loc = (message.getLineNumber() == 0 ? "" :
                    ("(" + message.getLineNumber() + "," +
                            message.getColumnNumber() + ")"));
                if (loc.length() == 0
                        && !message.getFileName().equals(message.getFullName())) {
                    loc = "(" + message.getFullName() + ")";
                }
                errorMessageBuilder.append(message.getFileName() + loc + ":" +
                        message.getProblem()).append('\n');
            }
            RunTestsResult rtr = deployDetails.getRunTestResult();
            if (rtr.getFailures() != null) {
                for (RunTestFailure failure : rtr.getFailures()) {
                    String n = (failure.getNamespace() == null ? "" :
                        (failure.getNamespace() + ".")) + failure.getName();
                    errorMessageBuilder.append("Test failure, method: " + n + "." +
                            failure.getMethodName() + " -- " +
                            failure.getMessage() + " stack " +
                            failure.getStackTrace() + "\n\n");
                }
            }
            if (rtr.getCodeCoverageWarnings() != null) {
                for (CodeCoverageWarning ccw : rtr.getCodeCoverageWarnings()) {
                    errorMessageBuilder.append("Code coverage issue");
                    if (ccw.getName() != null) {
                        String n = (ccw.getNamespace() == null ? "" :
                            (ccw.getNamespace() + ".")) + ccw.getName();
                        errorMessageBuilder.append(", class: " + n);
                    }
                    errorMessageBuilder.append(" -- " + ccw.getMessage() + "\n");
                }
            }
        }

        if (errorMessageBuilder.length() > 0) {
            errorMessageBuilder.insert(0, messageHeader);
            System.out.println(errorMessageBuilder.toString());
        }
    }

    private void createMetadataConnection(
        final String username,
        final String password,
        final String loginUrl) throws ConnectionException {

        final ConnectorConfig loginConfig = new ConnectorConfig();
        loginConfig.setAuthEndpoint(loginUrl);
        loginConfig.setServiceEndpoint(loginUrl);
        loginConfig.setManualLogin(true);
        LoginResult loginResult = (new EnterpriseConnection(loginConfig)).login(
                username, password);

        final ConnectorConfig metadataConfig = new ConnectorConfig();
        metadataConfig.setServiceEndpoint(loginResult.getMetadataServerUrl());
        metadataConfig.setSessionId(loginResult.getSessionId());
        this.metadataConnection = new MetadataConnection(metadataConfig);
    }

}
```

IN THIS SECTION:

1. Deleting Components from an Organization

   To delete components, perform a deployment with the `deploy()` call by using a destructive changes manifest file that lists the components to remove from your organization. You can perform a deployment that only deletes components, or a deployment that deletes and adds components. In API version 33.0 and later, you can specify components to delete before and after other components are added or updated. In earlier API versions, if deletions and additions are specified for the same deployment, the `deploy()` call performs the deletions first.

2. checkDeployStatus()

   Checks the status of declarative metadata call `deploy()`.

3. cancelDeploy()

   Cancels a deployment that hasn't completed yet.

SEE ALSO:

Running Tests in a Deployment

Deleting Components from an Organization

To delete components, perform a deployment with the `deploy()` call by using a destructive changes manifest file that lists the components to remove from your organization. You can perform a deployment that only deletes components, or a deployment that deletes and adds components. In API version 33.0 and later, you can specify components to delete before and after other components are added or updated. In earlier API versions, if deletions and additions are specified for the same deployment, the `deploy()` call performs the deletions first.

Deleting Components in a Deployment

To delete components, use the same procedure as with deploying components, but also include a delete manifest file that's named `destructiveChanges.xml` and list the components to delete in this manifest. The format of this manifest is the same as `package.xml` except that wildcards aren't supported.

Note: You can't use `destructiveChanges.xml` to delete items that are associated with an active Lightning page, such as a custom object, a component on the page, or the page itself. First, you must remove the page's action override by deactivating it in the Lightning App Builder.

The following sample `destructiveChanges.xml` file names a single custom object to be deleted:

```
<?xml version="1.0" encoding="UTF-8"?>
<Package xmlns="http://soap.sforce.com/2006/04/metadata">
    <types>
        <members>MyCustomObject__c</members>
        <name>CustomObject</name>
    </types>
</Package>
```

To deploy the destructive changes, you must also have a `package.xml` file that lists no components to deploy, includes the API version, and is in the same directory as `destructiveChanges.xml`:

```
<?xml version="1.0" encoding="UTF-8"?>
<Package xmlns="http://soap.sforce.com/2006/04/metadata">
    <version>48.0</version>
</Package>
```

Note:

- To bypass the Recycle Bin, set the purgeOnDelete option to true.
- When you delete a roll-up summary field using Metadata API, the field isn't saved in the Recycle Bin. The field is purged even if you don't set the purgeOnDelete deployment option to true.
- If you try to delete some components that don't exist in the organization, the rest of the deletions are still attempted.

Adding and Deleting Components in a Single Deployment

You can perform a deployment that specifies components to delete in destructiveChanges.xml and components to add or update in package.xml. The process is the same as with performing a delete-only deployment except that package.xml contains the components to add or update.

By default, deletions are processed before component additions. In API version 33.0 and later, you can specify components to be deleted before and after component additions. The process is the same as with performing a delete-only deployment except that the name of the deletion manifest file is different.

- To delete components *before* adding or updating other components, create a manifest file that's named destructiveChangesPre.xml and include the components to delete.
- To delete components *after* adding or updating other components, create a manifest file that's named destructiveChangesPost.xml and include the components to delete.

The ability to specify when deletions are processed is useful when you're deleting components with dependencies. For example, if a custom object is referenced in an Apex class, you can't delete it unless you modify the Apex class first to remove the dependency on the custom object. In this example, you can perform a single deployment that updates the Apex class to clear the dependency and then deletes the custom object by using destructiveChangesPost.xml. The following are samples of the package.xml and destructiveChangesPost.xml manifests that would be used in this example.

Sample package.xml, which specifies the class to update:

```xml
<?xml version="1.0" encoding="UTF-8"?>
<Package xmlns="http://soap.sforce.com/2006/04/metadata">
    <types>
        <members>SampleClass</members>
        <name>ApexClass</name>
    </types>
    <version>48.0</version>
</Package>
```

Sample destructiveChangesPost.xml, which specifies the custom object to delete after the class update:

```xml
<?xml version="1.0" encoding="UTF-8"?>
<Package xmlns="http://soap.sforce.com/2006/04/metadata">
    <types>
        <members>MyCustomObject__c</members>
        <name>CustomObject</name>
    </types>
</Package>
```

Note: The API version that the deployment uses is the API version that's specified in package.xml.

checkDeployStatus()

Checks the status of declarative metadata call `deploy()`.

Syntax

```
DeployResult = metadatabinding.checkDeployStatus(ID id, includeDetails boolean);
```

Usage

`checkDeployStatus` is used as part of the process for deploying packaged or unpackaged components to an organization:

1. Issue a `deploy()` call to start the asynchronous deployment. An AsyncResult object is returned. Note the value in the `id` field and use it for the next step.
2. Issue a `checkDeployStatus()` call in a loop until the `done` field of the returned DeployResult contains `true`, which means that the call is completed. The DeployResult object contains information about an in-progress or completed deployment started using the `deploy()` call. When calling `checkDeployStatus()`, pass in the `id` value from the AsyncResult object from the first step.

Note: Calls to `checkDeployStatus()` don't count against the API request limits and allocations.

In API version 29.0, Salesforce improved the deployment status properties and removed the requirement to use `checkStatus()` after a `deploy()` call to get information about deployments. Salesforce continues to support the use of `checkStatus()` when using `deploy()` with API version 28.0 or earlier.

Sample Code—Java

See the `deploy()` sample code for sample usage of this call.

Arguments

Name	Type	Description
`id`	ID	ID obtained from an AsyncResult object returned by `deploy()` or a subsequent `checkDeployStatus()` call.
`includeDetails`	boolean	Sets the DeployResult object to include DeployDetails information ((`true`) or not (`false`). The default is `false`. Available in API version 29.0 and later.

Response

DeployResult

cancelDeploy()

Cancels a deployment that hasn't completed yet.

Syntax

```
CancelDeployResult = metadatabinding.cancelDeploy(string id)
```

Usage

Use the `cancelDeploy()` operation to cancel a deployment in your organization started by the `deploy()` operation, which includes deployments started by the Lightning Platform Migration Tool and the Lightning Platform IDE. The deployment can be in a queue waiting to get started, or can be in progress. This operation takes the ID of the deployment you wish to cancel and returns a CancelDeployResult object. When the deployment is in the queue and hasn't started yet, calling `cancelDeploy()` cancels the deployment immediately. When the deployment has started and is in progress, it might not get canceled immediately, so you should call `checkDeployStatus()` to check the status of the cancellation.

Cancel a deployment using these steps.

1. Obtain the ID of the deployment you wish to cancel. For example, you can obtain the ID from the `deploy()` call in the `AsyncResult` object `id` field. Alternatively, you can obtain the ID in the Salesforce user interface from Setup by entering *`Deployment Status`* in the `Quick Find` box, selecting **Deployment Status**, and then noting the ID of a deployment started by the API.
2. Issue a `cancelDeploy()` call to start the cancellation process. This call returns a `CancelDeployResult` object.
3. Check the value in the `done` field of the returned `CancelDeployResult`. If the `done` field value is `true`, the deployment has been canceled and you're done. If the `done` field value is `false`, the cancellation is in progress, and follow these steps to check the cancellation status.
    a. Call `checkDeployStatus()` using the deployment ID you obtained earlier.
    b. In the returned DeployResult object, check the `status` field. If the status is `Canceling`, this means the cancellation is still in progress, and repeat steps a and b. Otherwise, if the status is `Canceled`, this means the deployment has been canceled and you're done.

The `deploy()` operation throws these API faults.

INVALID_ID_FIELD with the message `Invalid deploy ID`

The specified ID argument doesn't correspond to a valid deployment.

INVALID_ID_FIELD with the message `Deployment already completed`

The specified deployment has already completed.

Version

Available in API version 30.0 and later.

Permissions

Your client application must be logged in with the Modify All Data or Modify Metadata Through Metadata API Functions permission.

Note: If a user requires access to metadata but not to data, you can enable the Modify Metadata Through Metadata API Functions permission to give the access the user needs without providing access to org data.

Arguments

Name	Type	Description
id	string	The ID of the deployment to cancel.

Response

CancelDeployResult

Sample Code—Java

This sample shows how to cancel a deployment. The sample calls `cancelDeploy()` by passing it a given deployment ID. Next, it checks whether the cancellation has completed, and if not, calls `checkDeployStatus` in a loop.

```
public void cancelDeploy(String asyncId) throws Exception {
    // Issue the deployment cancellation request
    CancelDeployResult result = metadataConnection.cancelDeploy(asyncId);

    // If the deployment cancellation completed, write a message to the output.
    if (result.isDone()) {
        System.out.println("Your deployment was canceled successfully!");
    }
    else {
        // The deployment cancellation is still in progress, so get a new status
        DeployResult deployResult = metadataConnection.checkDeployStatus(asyncId, false);


        // Check whether the deployment is done. If not done, this means
        // that the cancellation is still in progress and the status is Canceling.

        while (!deployResult.isDone()) {
            // Assert that the deployment status is Canceling
            assert deployResult.getStatus() == DeployStatus.Canceling;
            // Wait 2 seconds
            Thread.sleep(2000);
            // Get the deployment status again
            deployResult = metadataConnection.checkDeployStatus(asyncId, false);
        }

        // The deployment is done. Write the status to the output.
        // (When the deployment is done, the cancellation should have completed
        // and the status should be Canceled. However, in very rare cases,
        // the deployment can complete before it is canceled.)
        System.out.println("Final deploy status = >" + deployResult.getStatus());
    }
}
```

deployRecentValidation()

Deploys a recently validated component set without running Apex tests.

Syntax

```
string = metadatabinding.deployRecentValidation(ID validationID)
```

Usage

Use `deployRecentValidation()` to deploy your components to production in less time by skipping the execution of Apex tests. Ensure that the following requirements are met before deploying a recent validation.

- The components have been validated successfully for the target environment within the last 10 days.
- As part of the validation, Apex tests in the target org have passed.
- Code coverage requirements are met.
  - If all tests in the org or all local tests are run, overall code coverage is at least 75%, and Apex triggers have some coverage.
  - If specific tests are run with the `RunSpecifiedTests` test level, each class and trigger that was deployed is covered by at least 75% individually.

This call is equivalent to performing a quick deployment of a recent validation on the Deployment Status page in the Salesforce user interface.

Before you call `deployRecentValidation()`, your organization must have a validation that was recently run. You can run a validation on a set of components by calling deploy() with the `checkOnly` property of the *`deployOptions`* parameter set to `true`. Note the ID that you obtained from the `deploy()` call. You'll use this ID for the `deployRecentValidation()` call in the next step.

After you've run a validation successfully, use these steps to quick-deploy the validation to the same target environment.

1. To start an asynchronous quick deployment, call `deployRecentValidation()` and pass it the ID of a recent validation. This ID is obtained from the previous deploy() call. The `deployRecentValidation()` call returns the ID of the quick deployment. Note this value. You'll use it in the next step.
2. Check for the completion of the call. This process is similar to that of deploy(). Issue a `checkDeployStatus()` call in a loop until the `done` field of the returned DeployResult contains `true`, which means that the call is completed. The DeployResult object contains information about an in-progress or completed deployment that was started by using the `deployRecentValidation()` call. When calling `checkDeployStatus()`, pass in the ID value that you obtained in the first step.

Version

Available in API version 33.0 and later.

Arguments

Name	Type	Description
`validationID`	string	The ID of a recent validation.

Response

Type: string

The ID of the quick deployment.

Sample Code—Java

```
package com.salesforce.test.metadata;

import java.rmi.RemoteException;

import com.sforce.soap.metadata.CodeCoverageWarning;
import com.sforce.soap.metadata.DeployDetails;
import com.sforce.soap.metadata.DeployMessage;
import com.sforce.soap.metadata.DeployResult;
import com.sforce.soap.metadata.MetadataConnection;
import com.sforce.soap.metadata.RunTestFailure;
import com.sforce.soap.metadata.RunTestsResult;
import com.sforce.soap.partner.Connector;
import com.sforce.ws.ConnectionException;
import com.sforce.ws.ConnectorConfig;

/**
 * Quick-deploy a recent validation.
 * Prerequisite: A successful validation (check-only deploy) has been done in the org
recently.
 */
public class DeployRecentValidationSample {
    // binding for the metadata WSDL used for making metadata API calls
    private MetadataConnection metadataConnection;

    // one second in milliseconds
    private static final long ONE_SECOND = 1000;
    // maximum number of attempts to deploy the zip file
    private static final int MAX_NUM_POLL_REQUESTS = 50;

    public static void main(String[] args) throws Exception {
        final String USERNAME = args[0];
        final String PASSWORD = args[1];
        final String URL = args[2];

        final String recentValidationId = args[3];

        DeployRecentValidationSample sample = new DeployRecentValidationSample(
                          USERNAME, PASSWORD, URL);
        sample.deployRecentValidation(recentValidationId);
    }

    public DeployRecentValidationSample(String username, String password, String loginUrl)

            throws ConnectionException {
        createMetadataConnection(username, password, loginUrl);
    }

    public void deployRecentValidation(String recentValidationId)
        throws RemoteException, Exception
```

```
{
   String asyncResultId = metadataConnection.deployRecentValidation(recentValidationId);


    // Wait for the deploy to complete
    int poll = 0;
    long waitTimeMilliSecs = ONE_SECOND;
    DeployResult deployResult = null;
    boolean fetchDetails;
    do {
        Thread.sleep(waitTimeMilliSecs);
        // double the wait time for the next iteration
        waitTimeMilliSecs *= 2;
        if (poll++ > MAX_NUM_POLL_REQUESTS) {
            throw new Exception("Request timed out. If this is a large set " +
                    "of metadata components, check that the time allowed by " +
                    "MAX_NUM_POLL_REQUESTS is sufficient.");
        }

        // Fetch in-progress details once for every 3 polls
        fetchDetails = (poll % 3 == 0);
      deployResult = metadataConnection.checkDeployStatus(asyncResultId, fetchDetails);

        System.out.println("Status is: " + deployResult.getStatus());
        if (!deployResult.isDone() && fetchDetails) {
            printErrors(deployResult, "Failures for deployment in progress:\n");
        }
    }
    while (!deployResult.isDone());

    if (!deployResult.isSuccess() && deployResult.getErrorStatusCode() != null) {
        throw new Exception(deployResult.getErrorStatusCode() + " msg: " +
                deployResult.getErrorMessage());
    }

    if (!fetchDetails) {
        // Get the final result with details if we didn't do it in the last attempt.
        deployResult = metadataConnection.checkDeployStatus(asyncResultId, true);
    }

    if (!deployResult.isSuccess()) {
        printErrors(deployResult, "Final list of failures:\n");
        throw new Exception("The files were not successfully deployed");
    }

    System.out.println("The recent validation " + recentValidationId +
                        " was successfully deployed");
}

/**
 * Print out any errors, if any, related to the deploy.
 * @param result - DeployResult
 */
private void printErrors(DeployResult result, String messageHeader)
```

```
    {
        DeployDetails deployDetails = result.getDetails();

        StringBuilder errorMessageBuilder = new StringBuilder();
        if (deployDetails != null) {
            DeployMessage[] componentFailures = deployDetails.getComponentFailures();
            for (DeployMessage message : componentFailures) {
                String loc = (message.getLineNumber() == 0 ? "" :
                    ("(" + message.getLineNumber() + "," +
                            message.getColumnNumber() + ")"));
                if (loc.length() == 0
                        && !message.getFileName().equals(message.getFullName())) {
                    loc = "(" + message.getFullName() + ")";
                }
                errorMessageBuilder.append(message.getFileName() + loc + ":" +
                        message.getProblem()).append('\n');
            }
            RunTestsResult rtr = deployDetails.getRunTestResult();
            if (rtr.getFailures() != null) {
                for (RunTestFailure failure : rtr.getFailures()) {
                    String n = (failure.getNamespace() == null ? "" :
                        (failure.getNamespace() + ".")) + failure.getName();
                    errorMessageBuilder.append("Test failure, method: " + n + "." +
                            failure.getMethodName() + " -- " +
                            failure.getMessage() + " stack " +
                            failure.getStackTrace() + "\n\n");
                }
            }
            if (rtr.getCodeCoverageWarnings() != null) {
                for (CodeCoverageWarning ccw : rtr.getCodeCoverageWarnings()) {
                    errorMessageBuilder.append("Code coverage issue");
                    if (ccw.getName() != null) {
                        String n = (ccw.getNamespace() == null ? "" :
                            (ccw.getNamespace() + ".")) + ccw.getName();
                        errorMessageBuilder.append(", class: " + n);
                    }
                    errorMessageBuilder.append(" -- " + ccw.getMessage() + "\n");
                }
            }
        }

        if (errorMessageBuilder.length() > 0) {
            errorMessageBuilder.insert(0, messageHeader);
            System.out.println(errorMessageBuilder.toString());
        }
    }

    private void createMetadataConnection(
        final String username,
        final String password,
        final String loginUrl) throws ConnectionException {

        final ConnectorConfig loginConfig = new ConnectorConfig();
        loginConfig.setUsername(username);
        loginConfig.setPassword(password);
        loginConfig.setAuthEndpoint(loginUrl);

        Connector.newConnection(loginConfig);


        final ConnectorConfig metadataConfig = new ConnectorConfig();
        metadataConfig.setServiceEndpoint(
            loginConfig.getServiceEndpoint().replace("/u/", "/m/"));
        metadataConfig.setSessionId(loginConfig.getSessionId());
        this.metadataConnection = com.sforce.soap.metadata.Connector.
            newConnection(metadataConfig);
    }

}
```

retrieve()

The `retrieve()` call retrieves XML file representations of components in an organization.

Syntax

```
AsyncResult = metadatabinding.retrieve(RetrieveRequest retrieveRequest)
```

Usage

Use this call to retrieve file representations of components in an organization.

Note: You can deploy or retrieve up to 10,000 files at once. AppExchange packages use different limits. In API version 43.0 and 44.0, AppExchange packages can contain up to 12,500 files. In API version 45.0, AppExchange packages can contain up to 17,500 files. In API version 46.0, AppExchange packages can contain up to 22,000 files. In API version 47.0 and later, AppExchange packages can contain up to 30,000 files. The maximum size of the deployed or retrieved .zip file is 39 MB. If the files are uncompressed in an unzipped folder, the size limit is 400 MB.

In API version 31.0 and later, the process of making a `retrieve()` call has been simplified. You no longer have to call `checkStatus()` after a `retrieve()` call to obtain the status of the retrieve operation. Instead, make calls to `checkRetrieveStatus()` only. If the retrieve operation is in progress, call `checkRetrieveStatus()` again until the retrieve operation is completed. The `checkStatus()` call is still supported in versions API version 30.0 or earlier, but is not available in API version 31.0 and later.

For API version 31.0 or later, retrieve packaged or unpackaged components by using the following steps.

1. Issue a `retrieve()` call to start the asynchronous retrieval. An AsyncResult object is returned. Note the value in the `id` field and use it for the next step.
2. Issue a `checkRetrieveStatus()` call and pass in the `id` value from the AsyncResult object from the first step. Check the value of the `done` field of the returned RetrieveResult. If it is `true`, this means that the call is completed and proceed to the next step. Otherwise, repeat this step to call `checkRetrieveStatus()` again until the `done` field is `true`.
3. Retrieve the zip file (`zipFile` field) and other desired fields from RetrieveResult that was returned by the final call to `checkRetrieveStatus()` in the previous step.

For API version 30.0 or earlier, retrieve packaged or unpackaged components by using the following steps.

1. Issue a `retrieve()` call to start the asynchronous retrieval. An AsyncResult object is returned. If the call is completed, the `done` field contains `true`. Most often, the call is not completed quickly enough to be noted in the result. If it is completed, note the value in the `id` field returned and skip the next step.
2. If the call is not complete, issue a `checkStatus()` call in a loop using the value in the `id` field of the AsyncResult object, returned by the `retrieve()` call in the previous step. Check the AsyncResult object returned until the `done` field contains `true`. The time taken to complete a `retrieve()` call depends on the size of the zip file being deployed, so use a longer wait time between iterations as the size of the zip file increases.
3. Issue a `checkRetrieveStatus()` call to obtain the results of the `retrieve()` call, using the `id` value returned in the first step.

For examples of manifest files, see Sample `package.xml` Manifest Files.

Permissions

Your client application must be logged in with the Modify All Data or Modify Metadata Through Metadata API Functions permission.

Note: If a user requires access to metadata but not to data, you can enable the Modify Metadata Through Metadata API Functions permission to give the access the user needs without providing access to org data.

Arguments

Name	Type	Description
`retrieveRequest`	RetrieveRequest	Encapsulates options for determining which packages or files are retrieved.

Response

AsyncResult

Sample Code—Java

This sample shows how to retrieve components into a zip file. See the `deploy()` sample code for details on how to deploy a zip file.

Note: This sample requires API version 34.0 or later.

```
package com.doc.samples;

import java.io.*;
import java.util.*;
import java.nio.ByteBuffer;
import java.nio.channels.*;
import java.rmi.RemoteException;
import javax.xml.parsers.DocumentBuilder;
import javax.xml.parsers.DocumentBuilderFactory;
import javax.xml.parsers.ParserConfigurationException;
import org.w3c.dom.Element;
import org.w3c.dom.Node;
import org.w3c.dom.NodeList;
import org.xml.sax.SAXException;

import com.sforce.soap.metadata.AsyncResult;
import com.sforce.soap.metadata.MetadataConnection;
import com.sforce.soap.enterprise.EnterpriseConnection;
import com.sforce.soap.metadata.RetrieveMessage;
import com.sforce.soap.metadata.RetrieveRequest;
import com.sforce.soap.metadata.RetrieveResult;
import com.sforce.soap.metadata.RetrieveStatus;
import com.sforce.soap.enterprise.LoginResult;
import com.sforce.ws.ConnectionException;
import com.sforce.ws.ConnectorConfig;
import com.sforce.soap.metadata.PackageTypeMembers;

public class RetrieveSample {

    // Binding for the metadata WSDL used for making metadata API calls
    private MetadataConnection metadataConnection;

    static BufferedReader rdr = new BufferedReader(new InputStreamReader(System.in));

    // one second in milliseconds
    private static final long ONE_SECOND = 1000;
    // maximum number of attempts to retrieve the results
    private static final int MAX_NUM_POLL_REQUESTS = 50;

    // manifest file that controls which components get retrieved
    private static final String MANIFEST_FILE = "package.xml";

    private static final double API_VERSION = 31.0;

    public static void main(String[] args) throws Exception {
        final String USERNAME = "user@company.com";
        // This is only a sample. Hard coding passwords in source files is a bad practice.

        final String PASSWORD = "password";
        final String URL = "https://login.salesforce.com/services/Soap/c/31.0";

        RetrieveSample sample = new RetrieveSample(USERNAME, PASSWORD, URL);
        sample.retrieveZip();
    }

    public RetrieveSample(String username, String password, String loginUrl)
            throws ConnectionException {
        createMetadataConnection(username, password, loginUrl);
    }


    private void retrieveZip() throws RemoteException, Exception
    {
        RetrieveRequest retrieveRequest = new RetrieveRequest();
        // The version in package.xml overrides the version in RetrieveRequest
        retrieveRequest.setApiVersion(API_VERSION);
        setUnpackaged(retrieveRequest);
        // Start the retrieve operation
        AsyncResult asyncResult = metadataConnection.retrieve(retrieveRequest);
        String asyncResultId = asyncResult.getId();

        // Wait for the retrieve to complete
        int poll = 0;
        long waitTimeMilliSecs = ONE_SECOND;
        RetrieveResult result = null;
        do {
            Thread.sleep(waitTimeMilliSecs);
            // Double the wait time for the next iteration
            waitTimeMilliSecs *= 2;
            if (poll++ > MAX_NUM_POLL_REQUESTS) {
                throw new Exception("Request timed out.  If this is a large set " +
                "of metadata components, check that the time allowed " +
                "by MAX_NUM_POLL_REQUESTS is sufficient.");
            }
            result = metadataConnection.checkRetrieveStatus(
                    asyncResultId, true);
            System.out.println("Retrieve Status: " + result.getStatus());
        } while (!result.isDone());

        if (result.getStatus() == RetrieveStatus.Failed) {
            throw new Exception(result.getErrorStatusCode() + " msg: " +
                    result.getErrorMessage());
        } else if (result.getStatus() == RetrieveStatus.Succeeded) {
            // Print out any warning messages
            StringBuilder buf = new StringBuilder();
            if (result.getMessages() != null) {
                for (RetrieveMessage rm : result.getMessages()) {
                    buf.append(rm.getFileName() + " - " + rm.getProblem());
                }
            }
            if (buf.length() > 0) {
                System.out.println("Retrieve warnings:\n" + buf);
            }

            // Write the zip to the file system
            System.out.println("Writing results to zip file");
            ByteArrayInputStream bais = new ByteArrayInputStream(result.getZipFile());
            File resultsFile = new File("retrieveResults.zip");
            FileOutputStream os = new FileOutputStream(resultsFile);
            try {
                ReadableByteChannel src = Channels.newChannel(bais);
                FileChannel dest = os.getChannel();
                copy(src, dest);

               System.out.println("Results written to " + resultsFile.getAbsolutePath());

            } finally {
                os.close();
            }
        }
```

```
    }

    /**
     * Helper method to copy from a readable channel to a writable channel,
     * using an in-memory buffer.
     */
    private void copy(ReadableByteChannel src, WritableByteChannel dest)
        throws IOException
    {
        // Use an in-memory byte buffer
        ByteBuffer buffer = ByteBuffer.allocate(8092);
        while (src.read(buffer) != -1) {
            buffer.flip();
            while(buffer.hasRemaining()) {
                dest.write(buffer);
            }
            buffer.clear();
        }
    }

    private void setUnpackaged(RetrieveRequest request) throws Exception
    {
        // Edit the path, if necessary, if your package.xml file is located elsewhere
        File unpackedManifest = new File(MANIFEST_FILE);
        System.out.println("Manifest file: " + unpackedManifest.getAbsolutePath());

        if (!unpackedManifest.exists() || !unpackedManifest.isFile())
            throw new Exception("Should provide a valid retrieve manifest " +
                    "for unpackaged content. " +
                    "Looking for " + unpackedManifest.getAbsolutePath());

        // Note that we populate the _package object by parsing a manifest file here.
        // You could populate the _package based on any source for your
        // particular application.
        com.sforce.soap.metadata.Package p = parsePackage(unpackedManifest);
        request.setUnpackaged(p);
    }

    private com.sforce.soap.metadata.Package parsePackage(File file) throws Exception {
        try {
            InputStream is = new FileInputStream(file);
            List<PackageTypeMembers> pd = new ArrayList<PackageTypeMembers>();
            DocumentBuilder db =
                DocumentBuilderFactory.newInstance().newDocumentBuilder();
            Element d = db.parse(is).getDocumentElement();
            for (Node c = d.getFirstChild(); c != null; c = c.getNextSibling()) {
                if (c instanceof Element) {
                    Element ce = (Element)c;
                    //
                    NodeList namee = ce.getElementsByTagName("name");
                    if (namee.getLength() == 0) {
                        // not
                        continue;
                    }
                    String name = namee.item(0).getTextContent();
                    NodeList m = ce.getElementsByTagName("members");
                    List<String> members = new ArrayList<String>();
                    for (int i = 0; i < m.getLength(); i++) {
                        Node mm = m.item(i);
                        members.add(mm.getTextContent());
                    }
                    PackageTypeMembers pdi = new PackageTypeMembers();
                    pdi.setName(name);
                    pdi.setMembers(members.toArray(new String[members.size()]));
                    pd.add(pdi);
                }
            }
            com.sforce.soap.metadata.Package r = new com.sforce.soap.metadata.Package();
            r.setTypes(pd.toArray(new PackageTypeMembers[pd.size()]));
            r.setVersion(API_VERSION + "");
            return r;
        } catch (ParserConfigurationException pce) {
            throw new Exception("Cannot create XML parser", pce);
        } catch (IOException ioe) {
            throw new Exception(ioe);
        } catch (SAXException se) {
            throw new Exception(se);
        }
    }


    private void createMetadataConnection(final String username,
            final String password, final String loginUrl)
            throws ConnectionException {

        final ConnectorConfig loginConfig = new ConnectorConfig();
        loginConfig.setAuthEndpoint(loginUrl);
        loginConfig.setServiceEndpoint(loginUrl);
        loginConfig.setManualLogin(true);
        LoginResult loginResult = (new EnterpriseConnection(loginConfig)).login(
                username, password);

        final ConnectorConfig metadataConfig = new ConnectorConfig();
        metadataConfig.setServiceEndpoint(loginResult.getMetadataServerUrl());
        metadataConfig.setSessionId(loginResult.getSessionId());
        this.metadataConnection = new MetadataConnection(metadataConfig);
    }

    //The sample client application retrieves the user's login credentials.
    // Helper function for retrieving user input from the console
    String getUserInput(String prompt) {
        System.out.print(prompt);
        try {
            return rdr.readLine();
        }
        catch (IOException ex) {
            return null;
        }
    }
}
```

RetrieveRequest

The RetrieveRequest parameter specified on a retrieve() call encapsulates options for determining which packages or files are retrieved.

The RetrieveRequest object consists of the following properties:

Name	Type	Description
apiVersion	double	Required. The API version for the retrieve request. The API version determines the fields retrieved for each metadata type. For example, an icon field was added to the CustomTab for API version 14.0. If you retrieve components for version 13.0 or earlier, the components will not include the icon field. Note: In API version 31.0 and later, the API version that's specified in package.xml is used for the retrieve() call and overrides the version in the apiVersion field. If the version is not specified in package.xml, the version in this field is used.
packageNames	string[]	A list of package names to be retrieved. If you are retrieving only unpackaged components, do not specify a name here. You can retrieve packaged and unpackaged components in the same retrieve.
singlePackage	boolean	Specifies whether only a single package is being retrieved (true) or not (false). If false, then more than one package is being retrieved.
specificFiles	string[]	A list of file names to be retrieved. If a value is specified for this property, packageNames must be set to null and singlePackage must be set to true.
unpackaged	Package	A list of components to retrieve that are not in a package.

checkRetrieveStatus()

Checks the status of the declarative metadata call retrieve() and returns the zip file contents.

Syntax

In API version 34.0 and later:

```
RetrieveResult = metadatabinding.checkRetrieveStatus(ID id, boolean includeZip);
```

In API version 33.0 and earlier:

```
RetrieveResult = metadatabinding.checkRetrieveStatus(ID id);
```

Usage

Use `checkRetrieveStatus()` to check the progress of the metadata `retrieve()` operation. The RetrieveResult object that this method returns indicates when the asynchronous `retrieve()` call is completed. If the retrieval is completed, RetrieveResult contains the zip file contents by default. Use the following process to retrieve metadata components with the `retrieve()` call.

1. Issue a `retrieve()` call to start the asynchronous retrieval. An AsyncResult object is returned. Note the value in the `id` field and use it for the next step.
2. Issue a `checkRetrieveStatus()` call and pass in the `id` value from the AsyncResult object from the first step. Check the value of the `done` field of the returned RetrieveResult. If it is `true`, this means that the call is completed and proceed to the next step. Otherwise, repeat this step to call `checkRetrieveStatus()` again until the `done` field is `true`.
3. Retrieve the zip file (`zipFile` field) and other desired fields from RetrieveResult that was returned by the final call to `checkRetrieveStatus()` in the previous step.

In API version 31.0 and later, the process of making a `retrieve()` call has been simplified. You no longer have to call `checkStatus()` after a `retrieve()` call to obtain the status of the retrieve operation. Instead, make calls to `checkRetrieveStatus()` only. If the retrieve operation is in progress, call `checkRetrieveStatus()` again until the retrieve operation is completed. The `checkStatus()` call is still supported in versions API version 30.0 or earlier, but is not available in API version 31.0 and later.

Retrieving the Zip File in a Second Process

By default, `checkRetrieveStatus()` returns the zip file on the last call to this operation when the retrieval is completed (`RetrieveResult.isDone() == true`) and then deletes the zip file from the server. Subsequent calls to `checkRetrieveStatus()` for the same retrieve operation can't retrieve the zip file after it has been deleted. Starting with API version 34.0, pass a boolean value for the `includeZip` argument of `checkRetrieveStatus()` to indicate whether to retrieve the zip file. The `includeZip` argument gives you the option to retrieve the file in a separate process after the retrieval operation is completed. For example, a service polls the retrieval status by calling `checkRetrieveStatus(id, false)` in a loop. This call returns the status of the retrieval operation, but doesn't retrieve the zip file. After the retrieval operation is completed, another process, such as a background file transfer service, calls `checkRetrieveStatus(id, true)` to retrieve the zip file. This last call causes the zip file to be deleted from the server.

```
// First process: Poll the retrieval but don't retrieve the zip file.
AsyncResult asyncResult = metadataConnection.retrieve(retrieveRequest);
String asyncResultId = asyncResult.getId();
// Wait for the retrieve to complete
int poll = 0;
long waitTimeMilliSecs = ONE_SECOND;
RetrieveResult result = null;
do {
    Thread.sleep(waitTimeMilliSecs);
    // Check the status but don't retrieve zip file.
    result = metadataConnection.checkRetrieveStatus(asyncResultId, false);
} while (!result.isDone());

// Second process: Retrieve the zip file.
// For example, this process can be a background file transfer service.
// Retrieve the zip file.
result = metadataConnection.checkRetrieveStatus(asyncResultId, true);
// Get the zip file from the RetrieveResult (result) variable
if (result.getStatus() == RetrieveStatus.Succeeded) {
    ByteArrayInputStream bais = new ByteArrayInputStream(result.getZipFile());
    // ...
}
```

Sample Code—Java

See the `retrieve()` sample code for sample usage of this call.

Arguments

Name	Type	Description
`id`	ID	ID obtained from an AsyncResult object returned by a `retrieve()` call or a subsequent RetrieveResult object returned by a `checkRetrieveStatus()` call.
`includeZip`	boolean	Set to `true` to retrieve the zip file. You can retrieve the zip file only after the retrieval operation is completed. After the zip file is retrieved, it is deleted from the server. Set to `false` to check the status of the retrieval without attempting to retrieve the zip file. If set to `null`, this argument defaults to `true`, which means that the zip file is retrieved on the last call to `checkRetrieveStatus()` when the retrieval has finished. This argument is available in API version 34.0 and later.

Response

RetrieveResult

CHAPTER 8 CRUD-Based Calls

Use CRUD-based calls to work with metadata components in a manner similar to how synchronous API calls in the enterprise WSDL act upon objects.

IN THIS SECTION:

createMetadata()

Adds one or more new metadata components to your organization synchronously.

readMetadata()

Returns one or more metadata components from your organization synchronously.

updateMetadata()

Updates one or more metadata components in your organization synchronously.

upsertMetadata()

Creates or updates one or more metadata components in your organization synchronously.

deleteMetadata()

Deletes one or more metadata components from your organization synchronously.

renameMetadata()

Renames a metadata component in your organization synchronously.

create()

Deprecated. Adds one or more new metadata components to your organization asynchronously. This call is removed as of API version 31.0 and is available in earlier versions only. Use `createMetadata()` instead.

delete()

Deprecated. Deletes one or more components from your organization asynchronously. This call is removed as of API version 31.0 and is available in earlier versions only. Use `deleteMetadata()` instead.

update()

Deprecated. Updates one or more components in your organization asynchronously. This call is removed as of API version 31.0 and is available in earlier versions only. Use `updateMetadata()` or `renameMetadata()` instead.

createMetadata()

Adds one or more new metadata components to your organization synchronously.

Syntax

```
SaveResult[] = metadatabinding.createMetadata(Metadata[] metadata);
```

Usage

Use the `createMetadata()` call to create any component that extends Metadata. All components must be of the same type in the same call. For more details, see Metadata Components and Types.

This call executes synchronously, which means that the call returns only when the operation completes.

Starting in API version 34.0, this call supports the AllOrNoneHeader header. By default, if `AllOrNoneHeader` isn't used in API version 34.0 and later, this call can save a partial set of records for records with no errors (equivalent to `AllOrNoneHeader=false`). In API version 33.0 and earlier, the default behavior is to only save all records when there are no failures in any record in the call (equivalent to `AllOrNoneHeader=true`).

Version

Available in API version 30.0 and later.

Permissions

Your client application must be logged in with the Modify All Data or Modify Metadata Through Metadata API Functions permission.

Note: If a user requires access to metadata but not to data, you can enable the Modify Metadata Through Metadata API Functions permission to give the access the user needs without providing access to org data.

Required Fields

Required fields are determined by the metadata components being created. For more information about specific component types, see Metadata Components and Types.

Valid Data Values

You must supply values that are valid for the field's data type, such as integers for integer fields (not alphabetic characters). In your client application, follow the data formatting rules specified for your programming language and development tool. (Your development tool handles the appropriate mapping of data types in SOAP messages.)

String Values

When storing values in string fields, the API trims any leading and trailing whitespace. For example, if the value of a `label` field is entered as `"MyObject "`, the value is stored in the database as `"MyObject"`.

Basic Steps for Creating Metadata Components

Use the following process to create metadata components:

1. Design an array and populate it with the components that you want to create. All components must be of the same type.
2. Call `createMetadata()` with the component array passed in as an argument.
3. A `SaveResult` object is returned for each component you tried to create. It contains information about whether the operation was successful, the name of the component created, and any errors returned if the operation wasn't successful.

Sample Code—Java

```
public void createCustomObjectSync() {
    try {
        CustomObject co = new CustomObject();
        String name = "MyCustomObject1";
        co.setFullName(name + "__c");
        co.setDeploymentStatus(DeploymentStatus.Deployed);
        co.setDescription("Created by the Metadata API");
        co.setEnableActivities(true);
        co.setLabel(name + " Object");
        co.setPluralLabel(co.getLabel() + "s");
        co.setSharingModel(SharingModel.ReadWrite);

        CustomField nf = new CustomField();
        nf.setType(FieldType.Text);
        nf.setLabel(co.getFullName() + " Name");
        co.setNameField(nf);

        SaveResult[] results = metadataConnection
                .createMetadata(new Metadata[] { co });

        for (SaveResult r : results) {
            if (r.isSuccess()) {
                System.out.println("Created component: " + r.getFullName());
            } else {
                System.out
                        .println("Errors were encountered while creating "
                                + r.getFullName());
                for (Error e : r.getErrors()) {
                    System.out.println("Error message: " + e.getMessage());
                    System.out.println("Status code: " + e.getStatusCode());
                }
            }
        }
    } catch (ConnectionException ce) {
        ce.printStackTrace();
    }
}
```

Arguments

Name	Type	Description
metadata	Metadata[]	Array of one or more metadata components. Limit: 10. (For CustomMetadata and CustomApplication only, the limit is 200.) You must submit arrays of only one type of component. For example, you can submit an array of 10 custom objects or 10 profiles, but not a mix of both types.

Response

SaveResult[]

`readMetadata()`

Returns one or more metadata components from your organization synchronously.

Syntax

```
ReadResult = metadataConnection.readMetadata(string metadataType, string[] fullNames);
```

Usage

Use the `readMetadata()` call to retrieve any component that extends Metadata. All components must be of the same type in the same call. For more details, see Metadata Components and Types.

This call executes synchronously, which means that the call returns only when the operation completes.

Version

Available in API version 30.0 and later.

Permissions

Your client application must be logged in with the Modify All Data or Modify Metadata Through Metadata API Functions permission.

Note: If a user requires access to metadata but not to data, you can enable the Modify Metadata Through Metadata API Functions permission to give the access the user needs without providing access to org data.

Basic Steps for Reading Metadata Components

Use the following process to read metadata components:

1. Determine the metadata type of the components you want to read, and the `fullName` of each component to read. See Metadata for more details on the `fullName` field. You can read only components of the same type in a single call.
2. Invoke the `readMetadata()` call. For the first argument, pass in the name of the metadata type. The metadata type must match one of the values returned by the `describeMetadata()` call. For the second argument, pass in an array of full names corresponding to the components you wish to get. The full names must match one or more full names returned by the `listMetadata()` call.
3. A `ReadResult` is returned that contains an array of `Metadata` components. Cast each returned `Metadata` object to the metadata type you specified in the call to get the component's properties.

Sample Code—Java

```
public void readCustomObjectSync() {
    try {
```

```
    ReadResult readResult = metadataConnection
            .readMetadata("CustomObject", new String[] {
                    "MyCustomObject1__c", "MyCustomObject2__c" });
    Metadata[] mdInfo = readResult.getRecords();
    System.out.println("Number of component info returned: "
            + mdInfo.length);
    for (Metadata md : mdInfo) {
        if (md != null) {
            CustomObject obj = (CustomObject) md;
            System.out.println("Custom object full name: "
                    + obj.getFullName());
            System.out.println("Label: " + obj.getLabel());
            System.out.println("Number of custom fields: "
                    + obj.getFields().length);
            System.out.println("Sharing model: "
                    + obj.getSharingModel());
        } else {
            System.out.println("Empty metadata.");
        }
    }
  } catch (ConnectionException ce) {
    ce.printStackTrace();
  }
}
```

Arguments

Name	Type	Description
`metadataType`	string	The metadata type of the components to read.
`fullNames`	string[]	Array of full names of the components to read. Limit: 10. (For CustomMetadata and CustomApplication only, the limit is 200.) You must submit arrays of only one type of component. For example, you can submit an array of 10 custom objects or 10 profiles, but not a mix of both types.

Response

ReadResult

updateMetadata()

Updates one or more metadata components in your organization synchronously.

Syntax

```
SaveResult[] = metadataConnection.updateMetadata(Metadata[] metadata);
```

Usage

Use the `updateMetadata()` call to update any component that extends Metadata. All components must be of the same type in the same call. For more details, see Metadata Components and Types.

This call executes synchronously, which means that the call returns only when the operation completes.

Starting in API version 34.0, this call supports the AllOrNoneHeader header. By default, if `AllOrNoneHeader` isn't used in API version 34.0 and later, this call can save a partial set of records for records with no errors (equivalent to `AllOrNoneHeader=false`). In API version 33.0 and earlier, the default behavior is to only save all records when there are no failures in any record in the call (equivalent to `AllOrNoneHeader=true`).

Version

Available in API version 30.0 and later.

Permissions

Your client application must be logged in with the Modify All Data or Modify Metadata Through Metadata API Functions permission.

Note: If a user requires access to metadata but not to data, you can enable the Modify Metadata Through Metadata API Functions permission to give the access the user needs without providing access to org data.

Required Fields

You must supply values for all the required fields in the component.

Valid Field Values

You must supply values that are valid for the field's data type, such as integers for integer fields (not alphabetic characters). In your client application, follow the data formatting rules specified for your programming language and development tool. (Your development tool handles the appropriate mapping of data types in SOAP messages.)

String Values

When storing values in string fields, the API trims any leading and trailing white space. For example, if the value of a `label` field is entered as `"MyObject "` the value is stored in the database as `"MyObject"`.

Basic Steps for Updating Metadata Components

Use this process to update metadata components:

1. Create an array of the components you wish to update. All components must be of the same type.
2. Invoke the `updateMetadata()` call, passing in the array of metadata components to update.

   A `SaveResult` object is returned for each component you tried to update. It contains information about whether the operation was successful, the name of the component updated, and any errors returned if the operation wasn't successful.

Sample Code—Java

```
public void updateCustomObjectSync() {
    try {
        CustomObject co = new CustomObject();
        String name = "MyCustomObject1";
        co.setFullName(name + "__c");
        co.setDeploymentStatus(DeploymentStatus.Deployed);
        co.setDescription("Updated description");
        co.setLabel(name + " Object Update");
        co.setPluralLabel(co.getLabel() + "s");
        co.setSharingModel(SharingModel.ReadWrite);

        // Name field with a type and label is required
        CustomField cf = new CustomField();
        cf.setType(FieldType.Text);
        cf.setLabel(co.getFullName() + " Name");
        co.setNameField(cf);

        SaveResult[] results = metadataConnection
                .updateMetadata(new Metadata[] { co });

        for (SaveResult r : results) {
            if (r.isSuccess()) {
                System.out.println("Updated component: " + r.getFullName());
            } else {
                System.out
                        .println("Errors were encountered while updating "
                                + r.getFullName());
                for (Error e : r.getErrors()) {
                    System.out.println("Error message: " + e.getMessage());
                    System.out.println("Status code: " + e.getStatusCode());
                }
            }
        }
    } catch (ConnectionException ce) {
        ce.printStackTrace();
    }
}
```

Arguments

Name	Type	Description
metadata	Metadata[]	Array of one or more metadata components you wish to update. Limit: 10. (For CustomMetadata and CustomApplication only, the limit is 200.) You must submit arrays of only one type of component. For example, you can submit an array of 10 custom objects or 10 profiles, but not a mix of both types.

Response

SaveResult[]

upsertMetadata()

Creates or updates one or more metadata components in your organization synchronously.

Syntax

```
UpsertResult[] = metadataConnection.upsertMetadata(Metadata[] metadata);
```

Usage

Use the `upsertMetadata()` call to create or update any component that extends Metadata. All components must be of the same type in the same call. For more details, see Metadata Components and Types.

If the specified components already exist in your organization, the `upsertMetadata()` call updates them. Otherwise, `upsertMetadata()` creates these components. Components are matched by the `fullname` field. This call executes synchronously, which means that the call returns only after the operation is completed.

Starting in API version 34.0, this call supports the AllOrNoneHeader header. By default, if `AllOrNoneHeader` isn't used in API version 34.0 and later, this call can save a partial set of records for records with no errors (equivalent to `AllOrNoneHeader=false`). In API version 33.0 and earlier, the default behavior is to only save all records when there are no failures in any record in the call (equivalent to `AllOrNoneHeader=true`).

Version

Available in API version 31.0 and later.

Permissions

Your client application must be logged in with the Modify All Data or Modify Metadata Through Metadata API Functions permission.

Note: If a user requires access to metadata but not to data, you can enable the Modify Metadata Through Metadata API Functions permission to give the access the user needs without providing access to org data.

Required Fields

You must supply values for all the required fields in the component.

Valid Field Values

You must supply values that are valid for the field's data type, such as integers (not alphabetic characters) for integer fields. In your client application, follow the data formatting rules that are specified for your programming language and development tool. (Your development tool handles the appropriate mapping of data types in SOAP messages.)

String Values

The API trims any leading and trailing white space when storing values in string fields. For example, if the value of a `label` field is entered as `" MyObject "`, the value is stored in the database as `"MyObject"`.

Basic Steps for Upserting Metadata Components

Use this process to upsert metadata components.

1. Create an array of Metadata objects that correspond to the components that you want to create or update. All components must be of the same type.
2. Invoke `upsertMetadata()`, passing in the array of metadata components that you created in the previous step.

   The `upsertMetadata()` call returns an array of `UpsertResult` objects. Each returned `UpsertResult` corresponds to a component that you upserted and contains information about the upsert operation—whether the operation was successful, the name of the component that was upserted, a flag indicating whether the component was created, and any errors that were returned if the operation wasn't successful.

Sample Code—Java

```
public void upsertMetadataSample() {
    try {
        // Create custom object to upsert
        CustomObject co = new CustomObject();
        String name = "MyCustomObject";
        co.setFullName(name + "__c");
        co.setDeploymentStatus(DeploymentStatus.Deployed);
        co.setDescription("Upserted by the Metadata API");
        co.setEnableActivities(true);
        co.setLabel(name + " Object");
        co.setPluralLabel(co.getLabel() + "s");
        co.setSharingModel(SharingModel.ReadWrite);

        CustomField nf = new CustomField();
        nf.setType(FieldType.Text);
        nf.setLabel("CustomField1");
        co.setNameField(nf);

        // Upsert the custom object
        UpsertResult[] results = metadataConnection
                .upsertMetadata(new Metadata[] { co });

        for (UpsertResult r : results) {
            if (r.isSuccess()) {
                System.out.println("Success!");
                if (r.isCreated()) {
                    System.out.println("Created component: "
                            + r.getFullName());
                } else {
                    System.out.println("Updated component: "
                            + r.getFullName());
                }
            } else {
                System.out
                .println("Errors were encountered while upserting "
                        + r.getFullName());
                for (Error e : r.getErrors()) {
                    System.out.println("Error message: " + e.getMessage());
                    System.out.println("Status code: " + e.getStatusCode());
                }
            }
        }
    } catch (ConnectionException ce) {
        ce.printStackTrace();
    }
}
```

Arguments

Name	Type	Description
`metadata`	Metadata[]	An array of one or more metadata components that you want to create or update Limit: 10. You must submit arrays of only one type of component. For example, you can submit an array of 10 custom objects or 10 profiles, but not a mix of both types.

Response

UpsertResult[]

deleteMetadata()

Deletes one or more metadata components from your organization synchronously.

Syntax

```
DeleteResult[] = metadataConnection.delete(string metadataType, string[] fullNames);
```

Usage

Use the `deleteMetadata()` call to delete any component that extends Metadata. All components must be of the same type in the same call. For more details, see Metadata Components and Types.

This call executes synchronously, which means that the call returns only when the operation completes.

Starting in API version 34.0, this call supports the AllOrNoneHeader header. By default, if the `AllOrNoneHeader` isn't used in any API version, this call can delete a partial set of records for records with no errors (equivalent to `AllOrNoneHeader=false`). If `AllOrNoneHeader` is set to `true`, no records are deleted if one or more records cause a failure.

Version

Available in API version 30.0 and later.

Permissions

Your client application must be logged in with the Modify All Data or Modify Metadata Through Metadata API Functions permission.

Note: If a user requires access to metadata but not to data, you can enable the Modify Metadata Through Metadata API Functions permission to give the access the user needs without providing access to org data.

Rules and Guidelines

When deleting components, consider the following rules and guidelines:

- Your client application must be logged in with sufficient access rights to delete individual components within the specified component. For more information, see "Factors that Affect Data Access" in the *SOAP API Developer Guide*.
- In addition, you might also need permission to access this component's parent component.
- To ensure referential integrity, this call supports cascading deletions. If you delete a parent component, you delete its children automatically, as long as each child component can be deleted.

Basic Steps for Deleting Metadata Components

Use the following process to delete metadata components:

1. Determine the metadata type of the components you want to delete and the `fullName` of each component to delete. You can delete only components of the same type in a single call. The full names must match one or more full names returned by the `listMetadata()` call. See Metadata for more details on the `fullName` field.
2. Invoke the `deleteMetadata()` call. For the first argument, pass in the name of the metadata type. For the second argument, pass in an array of full names corresponding to the components you wish to delete.

   A `DeleteResult` object is returned for each component you try to delete. It contains information about whether the operation was successful, the name of the deleted component, and any errors returned if the operation wasn't successful.

Sample Code—Java

```
public void deleteCustomObjectSync() {
    try {
        DeleteResult[] results = metadataConnection.deleteMetadata(
                "CustomObject", new String[] { "MyCustomObject1__c",
                        "MyCustomObject2__c" });
        for (DeleteResult r : results) {
            if (r.isSuccess()) {
                System.out.println("Deleted component: " + r.getFullName());
            } else {
                System.out
                        .println("Errors were encountered while deleting "
                                + r.getFullName());
                for (Error e : r.getErrors()) {
                    System.out.println("Error message: " + e.getMessage());
                    System.out.println("Status code: " + e.getStatusCode());
                }
            }
        }
    } catch (ConnectionException ce) {
        ce.printStackTrace();
    }
}
```

Arguments

Name	Type	Description
metadataType	string	The metadata type of the components to delete.
fullNames	string[]	Array of full names of the components to delete. Limit: 10. (For CustomMetadata and CustomApplication only, the limit is 200.) You must submit arrays of only one type of component. For example, you can submit an array of 10 custom objects or 10 profiles, but not a mix of both types.

Response

DeleteResult[]

renameMetadata()

Renames a metadata component in your organization synchronously.

Syntax

```
SaveResult = metadataConnection.renameMetadata(string metadataType, String oldFullname,
String newFullname);
```

Usage

Use the renameMetadata() call to rename one metadata component in your organization. This call executes synchronously, meaning the call returns only when the operation completes.

You can use this call to rename any of the objects that extend Metadata. For more details, see Metadata Components and Types.

Version

Available in API version 30.0 and later.

Permissions

Your client application must be logged in with the Modify All Data or Modify Metadata Through Metadata API Functions permission.

Note: If a user requires access to metadata but not to data, you can enable the Modify Metadata Through Metadata API Functions permission to give the access the user needs without providing access to org data.

Basic Steps for Renaming Metadata Components

Use the following process to rename a metadata component:

1. Determine the metadata type of the component you want to rename, its current full name, and the new full name. See Metadata for more details on the `fullName` field.
2. Invoke the `renameMetadata()` call. For the first argument, pass in the name of the metadata type. Pass in the old full name as the second argument and the new full name as the last argument.

   A `SaveResult` object is returned that contains information about whether the operation was successful, the name of the renamed component (which is the new name if the renaming was successful), and any errors returned if the operation wasn't successful.

Sample Code—Java

```
public void renameCustomObjectSync() {
 try {
    SaveResult[] results = metadataConnection.renameMetadata(
        "CustomObject", "MyCustomObject1__c","MyCustomObject1New__c");
    for (SaveResult r : results) {
     if (r.isSuccess()) {
      System.out.println("Renamed component: " + r.getName());
     }
     else {
      System.out.println("Errors were encountered while renaming " + r.getName());
      for(Error e : r.getErrors()) {
       System.out.println("Error message: " + e.getMessage());
       System.out.println("Status code: " + e.getStatusCode());
      }
     }
    }
  } catch (ConnectionException ce) {
    ce.printStackTrace();
  } catch (InterruptedException ie) {
    ie.printStackTrace();
  }
}
```

Arguments

Name	Type	Description
`metadataType`	string	The metadata type of the components to rename.
`oldFullName`	string	The current component full name.
`newFullName`	string	The new component full name.

Response

SaveResult

`create()`

Deprecated. Adds one or more new metadata components to your organization asynchronously. This call is removed as of API version 31.0 and is available in earlier versions only. Use `createMetadata()` instead.

Syntax

```
AsyncResult[] = metadatabinding.create(Metadata[] metadata);
```

Usage

Use this call to add one or more metadata components to your organization.

Version

This call is available in API version 30.0 and earlier only. This call is not available in API version 31.0 and later. Use `createMetadata()` instead.

Permissions

Your client application must be logged in with the "Modify All Data" permission.

Required Fields

Required fields are determined by the metadata components being created. For more information about specific component types, see Metadata Components and Types on page 135.

Valid Data Values

You must supply values that are valid for the field's data type, such as integers for integer fields (not alphabetic characters). In your client application, follow the data formatting rules specified for your programming language and development tool (your development tool handles the appropriate mapping of data types in SOAP messages).

String Values

When storing values in string fields, the API trims any leading and trailing whitespace. For example, if the value of a `label` field is entered as `"MyObject "` the value is stored in the database as `"MyObject"`.

Basic Steps for Creating Metadata Components

Use the following process to create metadata components:

1. Design an array and populate it with the components you want to create. All components must be of the same type.
2. Call `create()` with the component array passed in as an argument.
3. An AsyncResult object is returned for each component you triy to create, and is updated with status information as the operation moves from a queue to completed or error state. Call `checkStatus()` in a loop until the status values in AsyncResult indicate that all create operations are completed. Start with a wait time of one second between iterations of `checkStatus()` calls, and double the wait time each time you make a subsequent call.

Sample Code—Java

See Step 3: Walk Through the Java Sample Code on page 7 for sample Java code using the `create()` call.

Arguments

Name	Type	Description
`metadata`	Metadata[]	Array of one or more metadata components. Limit: 10. You must submit arrays of only one type of component. For example, you could submit an array of 10 custom objects or 10 profiles, but not a mix of both types.

Response

AsyncResult[]

SEE ALSO:

createMetadata()

update()

delete()

checkStatus()

`delete()`

Deprecated. Deletes one or more components from your organization asynchronously. This call is removed as of API version 31.0 and is available in earlier versions only. Use `deleteMetadata()` instead.

You can use this call to delete any of the objects that extend Metadata. For more details, see Metadata Components and Types on page 135.

Syntax

```
AsyncResult[] = metadataConnection.delete(Metadata[] metadata);
```

Usage

Use this call to delete one or more components from your organization.

Version

This call is available in API version 30.0 and earlier only. This call is not available in API version 31.0 and later. Use `deleteMetadata()` instead.

Permissions

Your client application must be logged in with the “Modify All Data” permission.

Rules and Guidelines

When deleting components, consider the following rules and guidelines:

- Your client application must be logged in with sufficient access rights to delete individual components within the specified component. For more information, see “Factors that Affect Data Access” in the *SOAP API Developer Guide*.
- In addition, you might also need permission to access this component’s parent component.
- To ensure referential integrity, this call supports cascading deletions. If you delete a parent component, you delete its children automatically, as long as each child component can be deleted.

Basic Steps for Deleting Metadata Components

Use the following process to delete metadata components:

1. Determine the `fullName` of each component you want to delete. See Metadata for more details on the `fullName` field. You can only delete components of the same type in a single call.
2. Invoke the `delete()` call, passing in the array of metadata components with `fullName` specified.
3. An AsyncResult object is returned for each component you try to delete, and is updated with status information as the operation moves from a queue to completed or error state. Call `checkStatus()` in a loop until the status values in AsyncResult indicate that all the delete operations are completed. Start with a wait time of one second between iterations of `checkStatus()` calls, and double the wait time each time you make a subsequent call.

Sample Code—Java

```
public void deleteCustomObject() {
  try {
    CustomObject co = new CustomObject();
    co.setFullName("MyCustomObject__c");
    AsyncResult[] ars = metadataConnection.create(new Metadata[]
       {co});
    AsyncResult asyncResult = ars[0];
    long waitTimeMilliSecs = 1000;
    while (!asyncResult.isDone()) {
      Thread.sleep(waitTimeMilliSecs);
      // double the wait time for the next iteration
      waitTimeMilliSecs *= 2;
      asyncResult = mdConnection.checkStatus(
        new String[] {asyncResult.getId()})[0];
      System.out.println("Status is: " + asyncResult.getState());
    }
  } catch (ConnectionException ce) {
    ce.printStackTrace();
  } catch (InterruptedException ie) {
    ie.printStackTrace();
  }
}
```

Arguments

Name	Type	Description
`metadata`	Metadata[]	Array of one or more metadata components. You only need to set the `fullName` field in the Metadata object. Limit: 10. You must submit arrays of only one type of component. For example, you could submit an array of 10 custom objects or 10 profiles, but not a mix of both types.

Response

AsyncResult[]

SEE ALSO:

deleteMetadata()

create()

update()

checkStatus()

`update()`

Deprecated. Updates one or more components in your organization asynchronously. This call is removed as of API version 31.0 and is available in earlier versions only. Use `updateMetadata()` or `renameMetadata()` instead.

This call can be used to update any of the objects that extend Metadata. For more details, see Metadata Components and Types on page 135.

Syntax

```
AsyncResult[] = metadataConnection.update(UpdateMetadata[] metadata);
```

Usage

Use this call to update one or more components. This call is analogous to the `ALTER TABLE` statement in SQL.

Version

This call is available in API version 30.0 and earlier only. This call is not available in API version 31.0 and later. Use `updateMetadata()` instead to update metadata components or `renameMetadata()` to rename a metadata component.

Permissions

Your client application must be logged in with the “Modify All Data” permission.

Required Fields

You must supply values for all the required fields in the component.

Valid Field Values

You must supply values that are valid for the field’s data type, such as integers for integer fields (not alphabetic characters). In your client application, follow the data formatting rules specified for your programming language and development tool (your development tool handles the appropriate mapping of data types in SOAP messages).

String Values

When storing values in string fields, the API trims any leading and trailing white space. For example, if the value of a `label` field is entered as `"MyObject "` the value is stored in the database as `"MyObject"`.

Basic Steps for Updating Metadata Components

Use this process to update metadata components:

1. Create an array of `UpdateMetadata` components and populate it with the components you wish to update. All components must be of the same type.
2. Invoke the `update()` call, passing in the array of metadata components to update.
3. An AsyncResult object is returned for each component you try to update, and is updated with status information as the operation moves from a queue to completed or error state. In a loop, call `checkStatus()` until the status values in AsyncResult indicate that all the update operations are completed. Start with a wait time of one second between iterations of `checkStatus()` calls, and double the wait time each time you make a subsequent call.

Sample Code—Java

```
public void updateCustomObject() {
  try {
    CustomObject co = new CustomObject();
    String name = "MyCustomObject";
    co.setFullName(name + "__c");
    co.setDeploymentStatus(DeploymentStatus.Deployed);
    co.setDescription("Created by the Metadata API");
    co.setEnableActivities(true);
    co.setLabel(name + " Object");
    co.setPluralLabel(co.getLabel() + "s");
    co.setSharingModel(SharingModel.ReadWrite);

    CustomField nf = new CustomField();
    nf.setType(FieldType.Text);
    nf.setLabel(co.getFullName() + " Name");

    co.setNameField(nf);

    UpdateMetadata updateMetadata = new UpdateMetadata();
    updateMetadata.setMetadata(co);
    updateMetadata.setCurrentName("TheCurrentName");

    AsyncResult[] ars = metadataConnection.update(new UpdateMetadata[]
        { updateMetadata });
    AsyncResult asyncResult = ars[0];
    // set initial wait time to one second in milliseconds
    long waitTimeMilliSecs = 1000;
    while (!asyncResult.isDone()) {
      Thread.sleep(waitTimeMilliSecs);
      // double the wait time for the next iteration
      waitTimeMilliSecs *= 2;
      asyncResult = metadataConnection.checkStatus(
        new String[] {asyncResult.getId()})[0];
      System.out.println("Status is: " + asyncResult.getState());
    }

    if (asyncResult.getState() != AsyncRequestState.Completed) {
      System.out.println(asyncResult.getStatusCode() + " msg: " +
          asyncResult.getMessage());
    }
  } catch (InterruptedException ie) {
    ie.printStackTrace();
  } catch (ConnectionException ce) {
    ce.printStackTrace();
  }
}
```

Arguments

Name	Type	Description
`metadata`	UpdateMetadata[]	Array of one or more UpdateMetadata data structures that represent the components you wish to update. Limit: 10. You must submit arrays of only one type of component. For example, you could submit an array of 10 custom objects or 10 profiles, but not a mix of both types.

UpdateMetadata

One or more UpdateMetadata objects are defined in the `metadata` argument. This object can be used to update any of the objects that extend Metadata. For more details, see Metadata Components and Types on page 135. Each UpdateMetadata object has the following fields:

Field	Field Type	Description
`currentName`	string	The API name of the component or field before the update. For example, if you wanted to update a CustomObject named Foo, the value of this field would be `Foo__c`. This value is supplied because this call may change the name, and the value here provides mapping.
`metadata`	Metadata	Full specification of the component or field you wish to update.

Response

AsyncResult[]

SEE ALSO:

- updateMetadata()
- create()
- delete()
- checkStatus()

CHAPTER 9 Utility Calls

Use utility calls to gather information that is useful for working with the file-based or CRUD-based calls.

- (Deprecated) `checkStatus()`
- `describeMetadata()`
- `describeValueType()`
- `listMetadata()`

`checkStatus()`

Deprecated. Checks the status of asynchronous metadata calls `create()`, `update()`, or `delete()`, or the declarative metadata call `retrieve()`. This call is removed as of API version 31.0 and is available only in earlier versions.

Note: Starting in API version 29.0, you no longer have to call `checkStatus()` after a `deploy()` call to get information about deployments. Similarly, starting in API version 31.0, you no longer have to call `checkStatus()` after a `retrieve()` call. The `checkStatus()` call has been replaced by checkDeployStatus() and checkRetrieveStatus() for deploy and retrieve operations respectively.

Syntax

```
AsyncResult[] = metadatabinding.checkStatus(ID[] ids);
```

Usage

Use this call to check whether or not an asynchronous metadata call or declarative metadata call has completed.

Version

This call is available only in API version 30.0 and earlier. This call is not available in API version 31.0 and later.

Sample Code—Java

See Step 3: Walk Through the Java Sample Code on page 7 for sample Java code using this call.

Arguments

Name	Type	Description
ids	ID[]	Array of one or more IDs. Each ID is returned in an AsyncResult and corresponds to a component being created, updated, deleted, deployed, or retrieved.

Response

AsyncResult[]

describeMetadata()

This call retrieves the metadata that describes your organization. This information includes Apex classes and triggers, custom objects, custom fields on standard objects, tab sets that define an app, and many other metadata types.

Syntax

```
DescribeMetadataResult = metadataConnection.describeMetadata(double apiVersion);
```

Arguments

Name	Type	Description
apiVersion	double	The API version for which you want metadata; for example, 48.0.

Permissions

Your client application must be logged in with the Modify All Data or Modify Metadata Through Metadata API Functions permission.

Note: If a user requires access to metadata but not to data, you can enable the Modify Metadata Through Metadata API Functions permission to give the access the user needs without providing access to org data.

Sample Code—Java

```
public void describeMetadata() {
  try {
    double apiVersion = 21.0;
    // Assuming that the SOAP binding has already been established.
    DescribeMetadataResult res =
        metadataConnection.describeMetadata(apiVersion);
    StringBuffer sb = new StringBuffer();
    if (res != null && res.getMetadataObjects().length > 0) {
      for (DescribeMetadataObject obj : res.getMetadataObjects()) {
        sb.append("***************************************************\n");
        sb.append("XMLName: " + obj.getXmlName() + "\n");
        sb.append("DirName: " + obj.getDirectoryName() + "\n");
        sb.append("Suffix: " + obj.getSuffix() + "\n");
        sb.append("****************************************************\n");
      }
    } else {
      sb.append("Failed to obtain metadata types.");
    }
    System.out.println(sb.toString());
  } catch (ConnectionException ce) {
    ce.printStackTrace();
  }
}
```

Response

DescribeMetadataResult

When to Use `describeMetadata()` and `describeValueType()`?

Use the `describeMetadata()` call to get high-level information about all the metadata types that are available for your organization, such as type names and file suffixes. Use the `describeValueType()` call to get granular information about a specific metadata type, such as fields contained within the type.

`describeValueType()`

Retrieves the metadata describing a given metadata type (value type).

`describeValueType()` accepts a namespace and a type name, and returns a `DescribeValueTypeResult` object. This call is available in API version 33.0 and later.

Syntax

```
DescribeValueTypeResult = connection.describeValueType("{namespace}type_name");
```

Example

Describe Apex class metadata in the Metadata namespace:

```
DescribeValueTypeResult =
metadataConnection.describeValueType("{http://soap.sforce.com/2006/04/metadata}ApexClass");
```

Describe Apex class metadata in the Tooling namespace:

```
DescribeValueTypeResult =
toolingConnection.describeValueType("{urn:metadata.tooling.soap.sforce.com}ApexClass");
```

Arguments

Name	Type	Description
`type`	string	The name of the metadata type for which you want metadata; for example, `ApexClass`. Include the namespace.

Permissions

Your client application must be logged in with the Modify All Data or Modify Metadata Through Metadata API Functions permission.

Note: If a user requires access to metadata but not to data, you can enable the Modify Metadata Through Metadata API Functions permission to give the access the user needs without providing access to org data.

Sample Code—Java

The following example describes several metadata types by specifying the Metadata namespace. Each metadata type is described using the helper method, `doDescribe()`, which calls the `describeValueType()` Metadata API call. The sample retrieves information from the returned `DescribeValueTypeResult`: a property, the parent field (if any), and the fields. Next, the sample iterates through the fields and outputs information about each field.

```
public void describeValueType() throws ConnectionException {
    doDescribe("{http://soap.sforce.com/2006/04/metadata}CustomObject");
    doDescribe("{http://soap.sforce.com/2006/04/metadata}CustomField");
    doDescribe("{http://soap.sforce.com/2006/04/metadata}EmailTemplate");
}

public void doDescribe(String type) throws ConnectionException {
    DescribeValueTypeResult result = metadataConnection.describeValueType(type);
    StringBuffer sb = new StringBuffer();

    sb.append("Describing " + type + " ...\n");

    if (result.getApiCreatable() == true) {
        sb.append("Is API creatable.\n");
    } else {
        sb.append("Is not API creatable.\n");
    }

    ValueTypeField parentField = result.getParentField();
    if (parentField != null) {
        sb.append("** Parent type fields **\n");
        if (parentField.getIsForeignKey()) {
            sb.append("This field is a foreign key.\n");
            for (String fkDomain : parentField.getForeignKeyDomain()) {
                sb.append("Foreign key domain: " + fkDomain + "\n");
            }
        }
    }

    sb.append("** Value type fields **\n");
    for(ValueTypeField field : result.getValueTypeFields()) {
        sb.append("****************************************************\n");
        sb.append("Name: " + field.getName() + "\n");
        sb.append("SoapType: " + field.getSoapType() + "\n");
        if (field.getIsForeignKey()) {
            sb.append("This field is a foreign key.\n");
            for (String fkDomain : field.getForeignKeyDomain()) {
                sb.append("Foreign key domain: " + fkDomain + "\n");
            }
        }
        sb.append("****************************************************\n");
    }
    System.out.println(sb.toString());
}
```

To run the previous example with the Tooling WSDL, replace the namespace with the Tooling namespace in the helper function call as follows. Also, use the Tooling connection instead of the Metadata connection to make the `describeValueType()` call.

```

    doDescribe("{urn:metadata.tooling.soap.sforce.com}CustomObject");
    doDescribe("{urn:metadata.tooling.soap.sforce.com}CustomField");
    doDescribe("{urn:metadata.tooling.soap.sforce.com}EmailTemplate");
```

After you run the sample, the output looks similar to the following.

```
Describing {http://soap.sforce.com/2006/04/metadata}CustomObject ...
Is API creatable.
** Value type fields **
****************************************************
Name: actionOverrides
SoapType: ActionOverride
****************************************************
****************************************************
Name: allowInChatterGroups
SoapType: boolean
****************************************************
****************************************************
Name: articleTypeChannelDisplay
SoapType: ArticleTypeChannelDisplay
****************************************************
****************************************************
Name: businessProcesses
SoapType: BusinessProcess
****************************************************
****************************************************
Name: compactLayoutAssignment
SoapType: string
****************************************************
****************************************************
Name: compactLayouts
SoapType: CompactLayout
****************************************************
****************************************************
Name: customHelp
SoapType: string
```

```
This field is a foreign key.
Foreign key domain: ApexPage
Foreign key domain: Scontrol
***************************************************
<The rest of the output for CustomObject has been omitted for brevity.>

Describing {http://soap.sforce.com/2006/04/metadata}CustomField ...
Is API creatable.
** Parent type fields **
This field is a foreign key.
Foreign key domain: CustomObject
** Value type fields **
***************************************************
Name: caseSensitive
SoapType: boolean
***************************************************
***************************************************

Name: defaultValue
SoapType: string
***************************************************

<The rest of the output has been omitted for brevity.>
```

Response

DescribeValueTypeResult

listMetadata()

This call retrieves property information about metadata components in your organization. Data is returned for the components that match the criteria specified in the queries parameter. The queries array can contain up to three `ListMetadataQuery` queries for each call. This call supports every metadata type: both top-level, such as CustomObject and ApexClass, and child types, such as CustomField and RecordType.

Syntax

```
FileProperties[] = metadataConnection.listMetadata(ListMetadataQuery[] queries, double
asOfVersion);
```

Usage

This call is useful when you want to identify individual components in `package.xml` for a `retrieve()` call or if you want a high-level view of particular metadata types in your organization. For example, you could use this call to return a list of names of all the CustomObject or Layout components in your organization, and use this information to make a subsequent `retrieve()` call to return a subset of these components. For more information about working with `package.xml`, see Deploying and Retrieving Metadata on page 16.

Note: This is a synchronous call so the results are returned in one call. This differs from asynchronous calls, such as `retrieve()`, where at least one subsequent call is needed to get the results.

Permissions

Your client application must be logged in with the Modify All Data or Modify Metadata Through Metadata API Functions permission.

Note: If a user requires access to metadata but not to data, you can enable the Modify Metadata Through Metadata API Functions permission to give the access the user needs without providing access to org data.

Sample Code—Java

The sample code below lists information about your custom objects. The code assumes that the SOAP binding has already been established.

```java
public void listMetadata() {
  try {
    ListMetadataQuery query = new ListMetadataQuery();
    query.setType("CustomObject");
    //query.setFolder(null);
    double asOfVersion = 48.0;
    // Assuming that the SOAP binding has already been established.
    FileProperties[] lmr = metadataConnection.listMetadata(
        new ListMetadataQuery[] {query}, asOfVersion);
    if (lmr != null) {
      for (FileProperties n : lmr) {
        System.out.println("Component fullName: " + n.getFullName());
        System.out.println("Component type: " + n.getType());
      }
    }
  } catch (ConnectionException ce) {
    ce.printStackTrace();
  }
}
```

Arguments

Name	Type	Description
`queries`	ListMetadataQuery[]	A list of objects that specify which components you are interested in.
`asOfVersion`	double	The API version for the metadata listing request. If you don't specify a value in this field, it defaults to the API version specified when you logged in. This field allows you to override the default and set another API version so that, for example, you could list the metadata for a metadata type that was added in a later version than the API version specified when you logged in. This field is available in API version 18.0 and later.

Response

FileProperties

ListMetadataQuery

The `ListMetadataQuery` parameter represents a list of objects that specify which components you are interested in.

Name	Type	Description
`folder`	string	The folder associated with the component. This field is required for components that use folders, such as Dashboard, Document, EmailTemplate, or Report.
`type`	string	Required. The metadata type, such as `CustomObject`, `CustomField`, or `ApexClass`.

CHAPTER 10 Result Objects

Use the following objects to get the results of your file-based or CRUD-based calls.

IN THIS SECTION:

AsyncResult

Contains the ID of a deployment or retrieval. In API version 28.0 and earlier, contains status information of any asynchronous metadata call.

CancelDeployResult

Contains information about a deployment cancellation—whether the cancellation completed and the deployment ID.

DeployResult

Contains information about the success or failure of the associated `deploy()` call.

DescribeMetadataResult

Contains information about the organization that is useful for developers working with declarative metadata.

DescribeValueTypeResult

Contains information about a value type that is useful for developers working with declarative metadata.

ReadResult

Contains result information for the `readMetadata` call.

RetrieveResult

Contains information about the success or failure of the associated `retrieve()` call.

SaveResult

Contains result information for the `createMetadata`, `updateMetadata`, or `renameMetadata` call.

DeleteResult

Contains result information for the `deleteMetadata` call.

UpsertResult

Contains information about the result of the associated `upsertMetadata()` call.

Error

Represents an error that occurred during a synchronous CRUD (`createMetadata()`, `updateMetadata()`, or `deleteMetadata()`) operation.

AsyncResult

Contains the ID of a deployment or retrieval. In API version 28.0 and earlier, contains status information of any asynchronous metadata call.

API Version 31.0 and Later

In API version 31.0, the process of retrieving metadata has been simplified and retrieval properties have been moved to RetrieveResult. Also, the asynchronous `create()`, `update()`, and `delete()` calls have been removed. Therefore, only the `id` field in AsyncResult is used. The `id` field is the ID of a deployment or retrieval.

AsyncResult is returned by the following asynchronous calls.

- `deploy()`
- `retrieve()`

AsyncResult has the following field that is in use.

Name	Type	Description
`id`	ID	Required. The ID of the component that's being deployed or retrieved.

All fields in AsyncResult other than `id` are deprecated as of API version 31.0. These fields exist but are no longer in use.

- `done`
- `message`
- `state`
- `statusCode`

API Versions 29.0 and 30.0

In API version 29.0, Salesforce moved several properties from the AsyncResult object to the DeployResult object and added several new ones, to improve the process for getting information about deployments. For more information about these changes, see `deploy()`.

In API versions 29.0 and 30.0, AsyncResult is returned by the same asynchronous calls as in API version 28.0 and earlier, but it has different fields.

Name	Type	Description
`done`	boolean	Required. Indicates whether the call has been completed (`true`) or not (`false`).
`id`	ID	Required. The ID of the component that's being created, updated, deleted, deployed, or retrieved.
`message`	string	The message that corresponds to the returned `statusCode` field, if any.
`state`	AsyncRequestState (enumeration of type string)	Required. The `AsyncRequestState` object has one of four possible values. • `Queued`: This call has not started. It is waiting in a queue. • `InProgress`: This call has started but has not been completed. • `Completed`: This call has been completed. • `Error`: An error occurred. See the `statusCode` for more information.

Name	Type	Description
statusCode	StatusCode (enumeration of type string)	If an error occurred during the create(), update(), or delete() call, a status code is returned, and the message that corresponds to the status code is returned in the message field. For a description of each StatusCode value, see "StatusCode" in the SOAP API Developer Guide.

API Version 28.0 and Earlier

In API version 28.0 and earlier, AsyncResult is returned by the following asynchronous calls.

- deploy()
- retrieve()
- create()
- update()
- delete()

Use the checkStatus() call against each object to discover when the call is completed for that object. Salesforce updates each AsyncResult object as the call is completed or when errors occur.

Similarly, the deploy() and retrieve() calls use AsyncResult, though you must subsequently use checkDeployStatus() or checkRetrieveStatus() respectively to get more status information for the deployment or retrieval.

AsyncResult has the following fields.

Name	Type	Description
checkOnly	boolean	Indicates whether this deployment is being used to check the validity of the deployed files without making any changes in the organization (true) or not (false). A check-only deployment does not deploy any components or change the organization in any way. This field is available in API version 16.0 and later and is relevant only for the deploy() call.
done	boolean	Required. Indicates whether the call has been completed (true) or not (false).
id	ID	Required. The ID of the component that's being created, updated, deleted, deployed, or retrieved.
message	string	The message that corresponds to the returned statusCode field, if any.
numberComponentErrors	int	The number of components that generated errors during this deployment. This field is available in API version 16.0 and later and is relevant only for the deploy() call.
numberComponentsDeployed	int	The number of components that have been deployed for this deployment. This field in conjunction with the numberComponentsTotal field gives you an indication of the progress of the deployment. This field is available in API version 16.0 and later and is relevant only for the deploy() call.

Name	Type	Description
numberComponentsTotal	int	The total number of components in the deployment. This field in conjunction with the numberComponentsDeployed field gives you an indication of the progress of the deployment. This field is available in API version 16.0 and later and is relevant only for the deploy() call.
numberTestErrors	int	The number of Apex tests that generated errors during this deployment. This field is available in API version 16.0 and later and is relevant only for the deploy() call.
numberTestsCompleted	int	The number of Apex tests that have been completed for this deployment. This field in conjunction with the numberTestsTotal field gives you an indication of the progress of tests for the deployment. This field is available in API version 16.0 and later and is relevant only for the deploy() call.
numberTestsTotal	int	The total number of Apex tests in the deployment. This field in conjunction with the numberTestsCompleted field gives you an indication of the progress of tests for the deployment. The value in this field is not accurate until the deployment has started running tests for the components that are being deployed. This field is available in API version 16.0 and later and is relevant only for the deploy() call.
secondsToWait	int	This field is no longer supported for API version 13.0 and later and is provided only for backward compatibility. The field was removed in API version 17.0. Indicates the number of seconds before the call is likel to be completed. This is an estimate only. A reasonable approach is to wait one second before calling checkStatus() to see if the operation is complete. Double your wait time for each successive iteration of checkStatus() calls until the operation is complete.
state	AsyncRequestState (enumeration of type string)	Required. The AsyncRequestState object has one of four possible values. • Queued: This call has not started. It is waiting in a queue. • InProgress: This call has started but has not been completed. • Completed: This call has been completed. • Error: An error occurred. See the statusCode for more information.
stateDetail	string	Indicates which component is being deployed or which Apex test class is running. This field is available in API version 16.0 and later and is relevant only for the deploy() call.
stateDetailLastModifiedDate	dateTime	The date and time when the stateDetail field was last modified. This field is available in API version 16.0 and later and is relevant only for the deploy() call.
statusCode	StatusCode (enumeration of type string)	If an error occurred during the create(), update(), delete(), or deploy() call, a status code is returned, and the message that corresponds to the status code is returned in the message field. For a description of each StatusCode value, see “StatusCode” in the SOAP API Developer Guide.

CancelDeployResult

Contains information about a deployment cancellation—whether the cancellation completed and the deployment ID.

The asynchronous metadata call `cancelDeploy()` returns a `CancelDeployResult` object.

Version

Available in API version 30.0 and later.

`CancelDeployResult` has the following properties.

Name	Type	Description
`done`	boolean	Indicates whether the deployment cancellation, which is started through `cancelDeploy()`, has completed (`true`) or not (`false`). When a deployment hasn't started yet and is still in the queue, the deployment is canceled immediately with the `cancelDeploy()` call and this field returns `true`. Otherwise, this field returns `false` when the cancellation is in progress.
`id`	ID	ID of the deployment being canceled.

DeployResult

Contains information about the success or failure of the associated `deploy()` call.

The asynchronous metadata call `checkDeployStatus()` returns a DeployResult object.

In API version 29.0, Salesforce moved several properties from the AsyncResult on page 101 object to the DeployResult object to improve the process for getting information about deployments. For more information about these changes, see `deploy()` on page 46.

For API version 29.0 and later, the DeployResult object has the following properties.

Name	Type	Description
`id`	ID	ID of the component being deployed.
`canceledBy`	ID	The ID of the user who canceled the deployment. This field is available in API version 30.0 and later.
`canceledByName`	string	The full name of the user who canceled the deployment. This field is available in API version 30.0 and later.
`checkOnly`	boolean	Indicates whether this deployment is being used to check the validity of the deployed files without making any changes in the organization (`true`) or not (`false`). A check-only deployment does not deploy any components or change the organization in any way.
`completedDate`	dateTime	Timestamp for when the deployment process ended.

Name	Type	Description
createdBy	ID	The ID of the user who created the deployment. This field is available in API version 30.0 and later.
createdByName	string	The full name of the user who created the deployment. This field is available in API version 30.0 and later.
createdDate	dateTime	Timestamp for when the deploy() call was received.
details	DeployDetails[]	Provides the details of a deployment that is in-progress or ended, if the includeDetails parameter is set to true in the checkDeployStatus() call.
done	boolean	Indicates whether the server finished processing the deploy() call for the specified id.
errorMessage	string	Message corresponding to the values in the errorStatusCode field, if any.
errorStatusCode	string	If an error occurred during the deploy() call, a status code is returned, and the message corresponding to the status code is returned in the errorMessagefield. For a description of each StatusCode value, see "StatusCode" in the *SOAP API Developer Guide*.
ignoreWarnings	boolean	Optional. Defaults to false. Specifies whether a deployment should continue even if the deployment generates warnings. Do not set this argument to true for deployments to production organizations.
lastModifiedDate	dateTime	Timestamp of the last update for the deployment process.
numberComponentErrors	int	The number of components that generated errors during this deployment.
numberComponentsDeployed	int	The number of components deployed in the deployment process. Use this value with the numberComponentsTotal value to get an estimate of the deployment's progress.
numberComponentsTotal	int	The total number of components in the deployment. Use this value with the numberComponentsDeployed value to get an estimate of the deployment's progress.
numberTestErrors	int	The number of Apex tests that have generated errors during this deployment.
numberTestsCompleted	int	The number of completed Apex tests for this deployment. Use this value with the numberTestsTotal value to get an estimate of the deployment's test progress.
numberTestsTotal	int	The total number of Apex tests for this deployment. Use this value with the numberTestsCompleted value to get an estimate of the deployment's test progress. The value in this field is not accurate until the deployment has started running tests for the components being deployed.

Name	Type	Description
runTestsEnabled	boolean	Indicates whether Apex tests were run as part of this deployment (true) or not (false). Tests are either automatically run as part of a deployment or can be set to run in DeployOptions for the deploy() call. For information on when tests are automatically run, see Running Tests in a Deployment. This field is available in API version 30.0 and later.
rollbackOnError	boolean	Optional. Defaults to true. Indicates whether any failure causes a complete rollback (true) or not (false). If false, whatever set of actions can be performed without errors are performed, and errors are returned for the remaining actions. This parameter must be set to true if you are deploying to a production organization.
startDate	dateTime	Timestamp for when the deployment process began.
stateDetail	string	Indicates which component is being deployed or which Apex test class is running.
status	DeployStatus (enumeration of type string)	Indicates the current state of the deployment. The valid values are: • Pending • InProgress • Succeeded • SucceededPartial • Failed • Canceling • Canceled
success	boolean	Indicates whether the deployment was successful (true) or not (false).

DeployDetails

These fields provide more information for the details field of the DeployResult object, if the includeDetails parameter is set to (true in the deploy() call.

Note: While a deployment is still in-progress, the DeployDetails object only contains componentFailures data. After the deployment process finishes, the other fields populate with the data for the entire deployment.

Name	Type	Description
componentFailures	DeployMessage[]	One or more DeployMessage objects containing deployment errors for each component.
componentSuccesses	DeployMessage[]	One or more DeployMessage objects containing successful deployment details for each component.
retrieveResult	RetrieveResult	If the performRetrieve parameter was specified for the deploy() call, a retrieve() call is performed immediately after the deploy() process completes. This field contains the results of that retrieval.

Name	Type	Description
runTestResult	RunTestsResult	If tests were run for the deploy() call, this field contains the test results. While a deployment is still in-progress, this field only contains error data. After the deployment process finishes, this field populates with the data for the entire deployment.

For API version 28.0 and earlier, the DeployResult object has the following properties.

Name	Type	Description
id	ID	ID of the component being deployed.
messages	DeployMessage[]	Contains information about the success or failure of a deploy() call.
retrieveResult	RetrieveResult	If the performRetrieve parameter was specified for the deploy() call, a retrieve() call is performed immediately after the deploy() process completes. This field contains the results of that retrieval.
runTestResult	RunTestsResult	If tests were run for the deploy() call, this field contains the test results.
success	boolean	Indicates whether the deployment was successful (true) or not (false).

DeployMessage

Each DeployResult object contains one or more DeployMessage objects.

Note: As of the Spring '20 release, only authenticated users can access DeployMessage objects.

Each DeployMessage object contains information about the deployment success or failure of a component in the deployment .zip file:

Name	Type	Description
changed	boolean	If true, the component was changed as a result of this deployment. If false, the deployed component was the same as the corresponding component already in the organization.
columnNumber	int	Each component is represented by a text file. If an error occurred during deployment, this field represents the column of the text file where the error occurred.
componentType	string	The metadata type of the component in this deployment. This field is available in API version 30.0 and later.
created	boolean	If true, the component was created as a result of this deployment. If false, the component was either deleted or modified as a result of the deployment.
createdDate	dateTime	The date and time when the component was created as a result of this deployment. This field is available in API version 30.0 and later.
deleted	boolean	If true, the component was deleted as a result of this deployment. If false, the component was either new or modified as result of the deployment.

Name	Type	Description
fileName	string	The name of the file in the .zip file used to deploy this component.
fullName	string	The full name of the component. Inherited from Metadata, this field is not defined in the WSDL for this metadata type. It must be specified when creating, updating, or deleting. See createMetadata() to see an example of this field specified for a call.
id	ID	ID of the component being deployed.
lineNumber	int	Each component is represented by a text file. If an error occurred during deployment, this field represents the line number of the text file where the error occurred.
problem	string	If an error or warning occurred, this field contains a description of the problem that caused the compile to fail.
problemType	DeployProblemType (enumeration of type string)	Indicates the problem type. The problem details are tracked in the problem field. The valid values are: • Warning • Error This field is available in API version 18.0 and later. Prior to version 18.0, there was no distinction between warnings and errors. All problems were treated as errors and prevented a successful deployment.
success	boolean	Indicates whether the component was successfully deployed (true) or not (false).

RunTestsResult

Contains information about the execution of unit tests, including whether unit tests were completed successfully, code coverage results, and failures.

A RunTestsResult object has the following properties

Name	Type	Description
apexLogId	string	The ID of an ApexLog object that is created at the end of a test run. The ApexLog object is created if there is an active trace flag on the user running an Apex test, or on a class or trigger being executed. This field is available in API version 35.0 and later.
codeCoverage	CodeCoverageResult[]	An array of one or more CodeCoverageResult objects that contains the details of the code coverage for the specified unit tests.
codeCoverageWarnings	CodeCoverageWarning[]	An array of one or more code coverage warnings for the test run. The results include both the total number of lines that could have been executed, as well as the number, line, and column positions of code that was not executed.

Name	Type	Description
failures	RunTestFailure[]	An array of one or more RunTestFailure objects that contain information about the unit test failures, if there are any.
flowCoverage	FlowCoverageResult[]	An array of results from test runs that executed flows. This field is available in API version 44.0 and later.
flowCoverageWarnings	FlowCoverageWarning[]	An array of warnings generated by test runs that executed flows. This field is available in API version 44.0 and later.
numFailures	int	The number of failures for the unit tests.
numTestsRun	int	The number of unit tests that were run.
successes	RunTestSuccess[]	An array of one or more RunTestSuccess objects that contain information about successes, if there are any.
totalTime	double	The total cumulative time spent running tests, in milliseconds. This can be helpful for performance monitoring.

CodeCoverageResult

The RunTestsResult object contains this object. It contains information about whether or not the compile of the specified Apex and run of the unit tests was successful.

Name	Type	Description
dmlInfo	CodeLocation[]	For each class or trigger tested, for each portion of code tested, this property contains the DML statement locations, the number of times the code was executed, and the total cumulative time spent in these calls. This can be helpful for performance monitoring.
id	ID	The ID of the CodeLocation. The ID is unique within an organization.
locationsNotCovered	CodeLocation[]	For each class or trigger tested, if any code is not covered, the line and column of the code not tested, and the number of times the code was executed.
methodInfo	CodeLocation[]	For each class or trigger tested, the method invocation locations, the number of times the code was executed, and the total cumulative time spent in these calls. This can be helpful for performance monitoring.
name	string	The name of the class or trigger covered.
namespace	string	The namespace that contained the unit tests, if one is specified.

Name	Type	Description
`numLocations`	int	The total number of code locations.
`soqlInfo`	CodeLocation[]	For each class or trigger tested, the location of SOQL statements in the code, the number of times this code was executed, and the total cumulative time spent in these calls. This can be helpful for performance monitoring.
`type`	string	Do not use. In early, unsupported releases, used to specify class or package.

CodeCoverageWarning

The RunTestsResult object contains this object. It contains information about the Apex class which generated warnings.

This object has the following properties.

Name	Type	Description
`id`	ID	The ID of the CodeLocation. The ID is unique within an organization.
`message`	string	The message of the warning generated.
`name`	string	The namespace that contained the unit tests, if one is specified.
`namespace`	string	The namespace that contained the unit tests, if one is specified.

RunTestFailure

The RunTestsResult object returns information about failures during the unit test run.

This object has the following properties.

Name	Type	Description
`id`	ID	The ID of the class which generated failures.
`message`	string	The failure message.
`methodName`	string	The name of the method that failed.
`name`	string	The name of the class that failed.
`namespace`	string	The namespace that contained the class, if one was specified.

Name	Type	Description
`seeAllData`	boolean	Indicates whether the test method has access to organization data (`true`) or not (`false`). This field is available in API version 33.0 and later.
`stackTrace`	string	The stack trace for the failure.
`time`	double	The time spent running tests for this failed operation, in milliseconds. This can be helpful for performance monitoring.
`type`	string	Do not use. In early, unsupported releases, used to specify class or package.

FlowCoverageResult

This object contains information about the flow version and the number of elements executed by the test run. This object is available in API version 44.0 and later.

Name	Type	Description
`elementsNotCovered`	string	List of elements in the flow version that weren't executed by the test run.
`flowId`	string	The ID of the flow version. The ID is unique within an org.
`flowName`	string	The name of the flow that was executed by the test run.
`flowNamespace`	string	The namespace that contains the flow, if one is specified.
`numElements`	int	The total number of elements in the flow version.
`numElementsNotCovered`	int	The number of elements in the flow version that weren't executed by the test run
`processType`	FlowProcessType (enumeration of type string)	The process type of the flow version.

FlowCoverageWarning

This object contains information about the flow version that generated warnings. This object is available in API version 44.0 and later.

Name	Type	Description
`flowId`	string	The ID of the flow version that generated the warning.
`flowName`	string	The name of the flow that generated the warning. If the warning applies to the overall test coverage of flows within your org, this value is null.
`flowNamespace`	string	The namespace that contains the flow, if one was specified.
`message`	string	The message of the warning that was generated.

RunTestSuccess

The RunTestsResult object returns information about successes during the unit test run.

This object has the following properties.

Name	Type	Description
`id`	ID	The ID of the class which generated the success.
`methodName`	string	The name of the method that succeeded.
`name`	string	The name of the class that succeeded.
`namespace`	string	The namespace that contained the unit tests, if one is specified.
`seeAllData`	boolean	Indicates whether the test method has access to organization data (`true`) or not (`false`). This field is available in API version 33.0 and later.
`time`	double	The time spent running tests for this operation. This can be helpful for performance monitoring.

CodeLocation

The RunTestsResult object contains this object in a number of fields.

This object has the following properties.

Name	Type	Description
`column`	int	The column location of the Apex tested.
`line`	int	The line location of the Apex tested.
`numExecutions`	int	The number of times the Apex was executed in the test run.
`time`	double	The total cumulative time spent at this location. This can be helpful for performance monitoring.

DescribeMetadataResult

Contains information about the organization that is useful for developers working with declarative metadata.

The `describeMetadata()` call returns a DescribeMetadataResult object.

Each DescribeMetadataResult object has the following properties:

Name	Type	Description
metadataObjects	DescribeMetadataObject[]	One or more metadata components and their attributes.
organizationNamespace	string	The namespace of the organization. Specify only for Developer Edition organizations that can contain a managed package. The managed package has a namespace specified when it is created.
partialSaveAllowed	boolean	Indicates whether rollbackOnError is allowed (true) or not (false). This value is always : • false in production organizations. • the opposite of testRequired.
testRequired	boolean	Indicates whether tests are required (true) or not (false). This value is always the opposite of partialSaveAllowed.

DescribeMetadataObject

This object is returned as part of the DescribeMetadataResult. Each DescribeMetadataObject has the following properties:

Name	Type	Description
childXmlNames	string[]	List of child sub-components for this component.
directoryName	string	The name of the directory in the .zip file that contains this component.
inFolder	boolean	Indicates whether the component is in a folder (true) or not (false). For example, documents, email templates and reports are stored in folders.
metaFile	boolean	Indicates whether the component requires an accompanying metadata file. For example, documents, classes, and s-controls are components that require an additional metadata file.
suffix	string	The file suffix for this component.
xmlName	string	The name of the root element in the metadata file for this component. This name also appears in the Packages > types > name field in the manifest file package.xml.

DescribeValueTypeResult

Contains information about a value type that is useful for developers working with declarative metadata.

The describeValueType() call returns a DescribeValueTypeResult object.

Each DescribeValueTypeResult object has the following properties.

Name	Type	Description
apiCreatable	boolean	Indicates whether this value type can be created through the createMetadata() call (true) or not (false). This field is available in API version 36.0 and later.
apiDeletable	boolean	Indicates whether this value type can be created through the deleteMetadata() call (true) or not (false). This field is available in API version 36.0 and later.
apiReadable	boolean	Indicates whether this value type can be created through the readMetadata() call (true) or not (false). This field is available in API version 36.0 and later.
apiUpdatable	boolean	Indicates whether this value type can be created through the updateMetadata() call (true) or not (false). This field is available in API version 36.0 and later.
parentField	ValueTypeField	Information about the parent of this value type. Parent field information is useful for metadata types that are specified with the parent in their name, such as custom fields, email templates, workflow rules, and reports. For example, the full name of a custom field includes the sObject that contains it (for example, Account.field1__c). Similarly, the full name of an email template includes the folder where the template is stored (for example, MyFolder/EmailTemplate1). If the value type has no parent, this field is null. This field is available in API version 36.0 and later.
valueTypeFields	ValueTypeField[]	One or more metadata components and their attributes.

ValueTypeField

This object is returned as part of the DescribeValueTypeResult and represents the metadata for one field. Each ValueTypeField has the following properties.

Name	Type	Description
fields	ValueTypeField	The ValueTypeField object for the next field, if any.
foreignKeyDomain	string	If isForeignKey is True, foreignKeyDomain is the type of object, such as Account or Opportunity.
isForeignKey	boolean	True if the field is a foreign key. That means this field is the primary key in a different database table.
isNameField	boolean	True if this value type field is a fullName field, otherwise False.
minOccurs	int	1 if this field is required, 0 otherwise.

Name	Type	Description
`name`	string	The name of this value type field. The name is null for parent fields.
`picklistValues`	PicklistEntry	The individual picklist values if the field is a picklist.
`soapType`	string	The data type of the field, such as boolean or double.
`valueRequired`	boolean	Required. Indicates whether this value type field must have a value (`true`) or can be null (`false`).

ReadResult

Contains result information for the `readMetadata` call.

Version

Available in API version 30.0 and later.

Properties

Name	Type	Description
`records`	Metadata[]	An array of metadata components returned from `readMetadata()`.

RetrieveResult

Contains information about the success or failure of the associated `retrieve()` call.

The metadata `retrieve()` call returns a RetrieveResult object.

Each RetrieveResult object has the following fields:

Name	Type	Description
`done`	boolean	Required. Indicates whether the `retrieve()` call is completed (`true`) or not (`false`). This field is available in API version 31.0 and later.
`errorMessage`	string	If an error occurs during the `retrieve()` call, this field contains a descriptive message about this error. This field is available in API version 31.0 and later.
`errorStatusCode`	StatusCode	If an error occurs during the `retrieve()` call, this field contains the status code for this error. This field is available in API version 31.0 and later. For a description of each StatusCode value, see “StatusCode” in the *SOAP API Developer Guide*.
`fileProperties`	FileProperties[]	Contains information about the properties of each component in the `.zip` file, and the manifest file `package.xml`. One object per component is returned.

Name	Type	Description
id	ID	ID of the component being retrieved.
messages	RetrieveMessage[]	Contains information about the success or failure of the retrieve() call.
status	RetrieveStatus (enumeration of type string)	The status of the retrieve() call. Valid values are: • Pending • InProgress • Succeeded • Failed This field is available in API version 31.0 and later.
success	boolean	Indicates whether the retrieve() call was successful (true) or not (false). This field is available in API version 31.0 and later.
zipFile	base64Binary	The zip file returned by the retrieve request. Base 64-encoded binary data. Prior to making an API call, client applications must encode the binary attachment data as base64. Upon receiving a response, client applications must decode the base64 data to binary. This conversion is usually handled for you by a SOAP client.

FileProperties

This component contains information about the properties of each component in the .zip file, and the manifest file package.xml. One object per component is returned. Note that this component does not contain information about any associated metadata files in the .zip file, only the component files and manifest file. FileProperties contains the following properties:

Name	Type	Description
createdById	string	Required. ID of the user who created the file.
createdByName	string	Required. Name of the user who created the file.
createdDate	dateTime	Required. Date and time when the file was created.
fileName	string	Required. Name of the file.
fullName	string	Required. The file developer name used as a unique identifier for API access. The value is based on the fileName but the characters allowed are more restrictive. The fullName can contain only underscores and alphanumeric characters. It must be unique, begin with a letter, not include spaces, not end with an underscore, and not contain two consecutive underscores.
id	string	Required. ID of the file.
lastModifiedById	string	Required. ID of the user who last modified the file.
lastModifiedByName	string	Required. Name of the user who last modified the file.
lastModifiedDate	dateTime	Required. Date and time that the file was last modified.

Name	Type	Description
`manageableState`	ManageableState (enumeration of type string)	Indicates the manageable state of the specified component if it is contained in a package: - `beta` - `deleted` - `deprecated` - `deprecatedEditable` - `installed` - `installedEditable` - `released` - `unmanaged`
`namespacePrefix`	string	If any, the namespace prefix of the component.
`type`	string	Required. The metadata type, such as `CustomObject`, `CustomField`, or `ApexClass`.

RetrieveMessage

RetrieveResult returns this object, which contains information about the success or failure of the `retrieve()` call. One object per problem is returned:

Name	Type	Description
`fileName`	string	The name of the file in the retrieved `.zip` file where a problem occurred.
`problem`	string	A description of the problem that occurred.

SEE ALSO:

retrieve()

SaveResult

Contains result information for the `createMetadata`, `updateMetadata`, or `renameMetadata` call.

Version

Available in API version 30.0 and later.

Properties

Name	Type	Description
`errors`	Error[]	An array of errors returned if the operation wasn't successful.
`fullName`	string	The full name of the component processed.
`success`	boolean	Indicates whether the operation was successful (`true`) or not (`false`).

DeleteResult

Contains result information for the `deleteMetadata` call.

Version

Available in API version 30.0 and later.

Properties

Name	Type	Description
`errors`	Error[]	An array of errors returned if the operation wasn't successful.
`fullName`	string	The full name of the deleted component.
`success`	boolean	Indicates whether the deletion was successful (`true`) or not (`false`).

UpsertResult

Contains information about the result of the associated `upsertMetadata()` call.

Version

Available in API version 31.0 and later.

Properties

Name	Type	Description
`created`	boolean	Indicates whether the upsert operation resulted in the creation of the component (`true`) or not (`false`). If `false` and the upsert operation was successful, this means that the component was updated.
`errors`	Error[]	An array of errors that were returned if the operation wasn't successful.

Name	Type	Description
`fullName`	string	The full name of the component that was created or updated if the operation was successful.
`success`	boolean	Indicates whether the operation was successful (`true`) or not (`false`).

Error

Represents an error that occurred during a synchronous CRUD (`createMetadata()`, `updateMetadata()`, or `deleteMetadata()`) operation.

Version

Available in API version 30.0 and later.

Properties

Name	Type	Description
`extendedErrorDetails`	ExtendedErrorDetails	More details about the error, including an extended error code and extra error properties, when available. Reserved for future use. For a description of the ExtendedErrorDetails element, see "ExtendedErrorDetails" in the *SOAP API Developer Guide*.
`fields`	string[]	An array containing names of fields that affected the error condition.
`message`	string	The error message text.
`statusCode`	StatusCode	A status code corresponding to the error. For a description of each StatusCode value, see "StatusCode" in the *SOAP API Developer Guide*.

CHAPTER 11 Metadata Types

Metadata API enables you to access some entities and feature settings that you can customize in the user interface.

Note:

- Metadata type names are case-sensitive. Specifying a type name with an invalid case results in a deployment error.
- Metadata types don't always correspond directly to their related data types. In some cases, the information is accessible but not organized as expected. For example, dependent picklists are exposed as a type of picklist, not a separate metadata type.
- The wildcard character doesn't apply to metadata types for feature settings, like AccountSettings. The wildcard applies only when retrieving all settings and not an individual setting. See Settings.

IN THIS SECTION:

Metadata Coverage Report

Launch the Metadata Coverage report to determine supported metadata components. The Metadata Coverage report is the ultimate source of truth for metadata coverage across several channels. These channels include Metadata API, scratch org source tracking, unlocked packages, second-generation managed packages, classic managed packages, and more.

Unsupported Metadata Types

Some things you can customize in a Salesforce org aren't available in Metadata API.

Special Behavior in Metadata API Deployments

Important considerations for specific types and contents of a deployment.

AccountRelationshipShareRule

Represents the rule that determines which object records are shared, how they are shared, the account relationship type that shares the records, and the level of access granted to the records. This type extends the MetadataWithContent metadata type and inherits its `content` and `fullName` fields.

ActionLinkGroupTemplate

Represents the action link group template. Action link templates let you reuse action link definitions and package and distribute action links. An action link is a button on a feed element. Clicking on an action link can take a user to another Web page, initiate a file download, or invoke an API call to an external server or Salesforce. Use action links to integrate Salesforce and third-party services into the feed. Every action link belongs to an action link group and action links within the group are mutually exclusive.

ActionPlanTemplate

Represents the instance of an action plan template.This type extends the Metadata metadata type and inherits its `fullName` field.

AnalyticSnapshot

Represents a reporting snapshot. A reporting snapshot lets you report on historical data. Authorized users can save tabular or summary report results to fields on a custom object, then map those fields to corresponding fields on a target object. They can then schedule when to run the report to load the custom object's fields with the report's data. Reporting snapshots enable you to work with report data similarly to how you work with other records in Salesforce.

AnimationRule

Represents criteria for determining when an animation is displayed to Path users.This type extends the Metadata metadata type and inherits its `fullName` field.

ArticleType

Represents the metadata associated with an article type.

ApexClass

Represents an Apex class. An Apex class is a template or blueprint from which Apex objects are created. Classes consist of other classes, user-defined methods, variables, exception types, and static initialization code.

ApexComponent

Represents a Visualforce component.

ApexPage

Represents a Visualforce page.

ApexTestSuite

Represents a suite of Apex test classes to include in a test run.

ApexTrigger

Represents an Apex trigger. A trigger is Apex code that executes before or after specific data manipulation language (DML) events occur, such as before object records are inserted into the database, or after records have been deleted.

AppMenu

Represents the app menu or the Salesforce mobile navigation menu.

AppointmentSchedulingPolicy

Represents a set of rules for scheduling appointments using Lightning Scheduler. This type extends the Metadata metadata type and inherits its `fullName` field.

ApprovalProcess

Represents the metadata associated with an approval process. An approval process automates how records are approved in Salesforce. An approval process specifies each step of approval, including who to request approval from and what to do at each point of the process.

AssignmentRules

Represents assignment rules that allow you to automatically route cases to the appropriate users or queues. You can access rules metadata for all applicable objects, for a specific object, or for a specific rule on a specific object.

AssistantContextItem (Beta)

AssistantContextItem is for internal use only.

AssistantDefinition (Beta)

AssistantDefinition and its child AssistantDefinitionProfile are for internal use only.

AssistantSkillQuickAction (Beta)

AssistantSkillQuickAction and its child AssistantSkillQuickActionParam are for internal use only.

AssistantSkillSobjectAction (Beta)

AssistantSkillSobjectAction and its child AssistantSkillSobjectParam are for internal use only.

AssistantVersion (Beta)

AssistantVersion and its children AssistantSkill, AssistantSkillIntent, and AssistantVersionAction are for internal use only.

Audience

Represents the audience in a community. An audience consists of different types of criteria, where the audience can be assigned and used for targeting in a community. This type extends the Metadata metadata type and inherits its `fullName` field.

AuraDefinitionBundle

Represents an Aura definition bundle. A bundle contains an Aura definition, such as an Aura component, and its related resources, such as a JavaScript controller. The definition can be a component, application, event, interface, or a tokens collection.

AuthProvider

Represents an authentication provider (auth provider). An auth provider lets users log in to your Salesforce org from an external service provider, such as Facebook, Google, or GitHub. This type extends the Metadata metadata type and inherits its `fullName` field.

AutoResponseRules

Represents an auto-response rule that sets conditions for sending automatic email responses to lead or case submissions based on the attributes of the submitted record. You can access rules metadata for all applicable objects, for a specific object, or for a specific rule on a specific object.

Bot

Represents a definition of an Einstein Bot configuration that can have one or more versions. Only one version can be active. This type extends the Metadata metadata type and inherits its `fullName` field.

BotVersion

Represents the configuration details for a specific Einstein Bot version, including dialogs, intents, entities, and variables. This type extends the Metadata metadata type and inherits its `fullName` field.

BrandingSet

Represents the definition of a set of branding properties for a Lightning community, as defined in the Theme panel in Experience Builder. This type extends the Metadata metadata type and inherits its `fullName` field.

CallCenter

Represents the Call Center definition used to integrate Salesforce with a third-party computer-telephony integration (CTI) system.

CampaignInfluenceModel

Represents a campaign influence model used by Customizable Campaign Influence. You can't configure Customizable Campaign Influence via the Metadata API, but you can add a campaign influence model.

CaseSubjectParticle

Represents the Social Business Rules custom format for the **Case Subject** field on cases created from inbound social posts.

Certificate

Represents a certificate used for digital signatures which verify that requests are coming from your org. Certificates are used for either authenticated single sign-on with an external website, or when using your org as an identity provider. This type extends the MetadataWithContent metadata type and inherits its `content` and `fullName` fields.

ChatterExtension

Represents the metadata used to describe a Rich Publisher App that's integrated with the Chatter publisher.

CleanDataService

Represents a data service that adds and updates data in standard objects. This type extends the Metadata metadata type and inherits its `fullName` field.

CMSConnectSource

Represents the connection information for external content management systems that feed content to Communities. This type extends the Metadata metadata type and inherits its `fullName` field.

Community (Zone)

Represents a zone that contains Ideas or Chatter Answers objects. Zones are shared by the Ideas, Answers, and Chatter Answers features, allowing you to view and create zones from those locations.This type extends the Metadata metadata type and inherits its `fullName` field.

CommunityTemplateDefinition

Represents the definition of a community template. This type extends the Metadata metadata type and inherits its `fullName` field.

CommunityThemeDefinition

Represents the definition of a community theme.This type extends the Metadata metadata type and inherits its `fullName` field.

ConnectedApp

Represents a connected app configuration. A connected app integrates an application with Salesforce using APIs. Connected apps use standard SAML and OAuth protocols to authenticate, provide single sign-on, and provide tokens for use with Salesforce APIs. In addition to standard OAuth capabilities, Salesforce admins set various security policies for connected apps and have explicit control over who can use the corresponding apps.

ContentAsset

Represents the metadata for creating an asset file. Asset files enable a Salesforce file to be used for org setup and configuration purposes. This type extends the MetadataWithContent metadata type and inherits its `content` and `fullName` fields.

CorsWhitelistOrigin

Represents an origin in the CORS whitelist.

CspTrustedSite

Represents a CSP Trusted Site. The Lightning Component framework uses Content Security Policy (CSP) to impose restrictions on content. The main objective is to help prevent cross-site scripting (XSS) and other code injection attacks. To use third-party APIs that make requests to an external (non-Salesforce) server or to use a WebSocket connection, add a CSP Trusted Site. This type extends the Metadata metadata type and inherits its `fullName` field.

CustomApplication

CustomApplication represents a custom or standard application. In API version 29.0 and earlier, CustomApplication represents only a custom application. An application is a list of tab references, with a description and a logo. This type extends the Metadata metadata type and inherits its `fullName` field.

CustomApplicationComponent

Represents a custom console component (Visualforce page) assigned to a CustomApplication that is marked as a Salesforce console. Custom console components extend the capabilities of Salesforce console apps. See “Customize a Console with Custom Components in Salesforce Classic” in the Salesforce online help.

CustomFeedFilter

Represents a custom feed filter that limits the feed view to feeds from the Cases object. The custom feed filter shows only feed items that satisfy the criteria specified in the CustomFeedFilter definition.This type extends the Metadata metadata type and inherits its `fullName` field.

CustomHelpMenuSection

Represents the section of the Lightning Experience help menu that the admin added to display custom, org-specific help resources for the org. The custom section contains help resources added by the admin. This type extends the Metadata metadata type and inherits its `fullName` field.

CustomLabels

The CustomLabels metadata type allows you to create custom labels that can be localized for use in different languages, countries, and currencies.

Custom Metadata Types (CustomObject)

Represents the metadata associated with a custom metadata type.

CustomNotificationType

Represents the metadata associated with a custom notification type. Custom notification types allow you to send a custom desktop or mobile notification via a process or invocable API call.

CustomObject

Represents a custom object that stores data unique to your org or an external object that maps to data stored outside your org.This type extends the Metadata metadata type and inherits its `fullName` field.

CustomObjectTranslation

This metadata type allows you to translate custom objects for a variety of languages.

CustomPageWebLink

Represents a custom link defined in a home page component. This type extends the Metadata metadata type and inherits its `fullName` field.

CustomPermission

Represents a permission that grants access to a custom feature. This type extends the Metadata metadata type and inherits its `fullName` field.

CustomSite

Represents a Salesforce site. Create public websites and applications that are directly integrated with your Salesforce organization, but don't require users to log in with a username and password.

CustomTab

Represents a custom tab. Custom tabs let you display custom object data or other web content in Salesforce. When you add a custom tab to an app in Salesforce Classic, it displays as a tab. When you add a custom tab to an app in Lightning Experience, it displays as an item in the app's navigation bar. When a tab displays a custom object, the tab name is the same as the custom object name; for page, s-control, or URL tabs, the name is arbitrary.

CustomValue

Represents the definition of a value used in a global value set or local custom picklist. Custom picklist fields can be local and unique, or can inherit their values from a global picklist (called a *global value set* in API version 38.0). This type extends the Metadata metadata type and inherits its `fullName` field.

Dashboard

Represents a dashboard. Dashboards are visual representations of data that allow you to see key metrics and performance at a glance.

DataCategoryGroup

Represents a data category group.

DelegateGroup

Represents a group of users who have the same administrative privileges. These groups are different from public groups used for sharing.

Document

Represents a Document. All documents must be in a document folder, such as `sampleFolder/TestDocument`.

DuplicateRule

Represents a rule that specifies how duplicate records in an object are detected. This type extends the Metadata metadata type and inherits its `fullName` field.

EclairGeoData

Represents an Analytics custom map chart. Custom maps are user-defined maps that are uploaded to Analytics and are used just as standard maps are. Custom maps are accessed in Analytics from the list of maps available with the map chart type.

EmailServicesFunction

Represents an email service. This type extends the Metadata metadata type and inherits its `fullName` field.

EmailTemplate

Represents an email template.

EmbeddedServiceBranding

Represents the branding for each Embedded Service deployment. This type extends the Metadata metadata type and inherits its `fullName` field.

EmbeddedServiceConfig

Represents a setup node for creating an Embedded Service for Web deployment. This type extends the Metadata metadata type and inherits its `fullName` field.

EmbeddedServiceFieldService

Represents a setup node for creating an embedded Appointment Management deployment. This type extends the Metadata metadata type and inherits its `fullName` field.

EmbeddedServiceFlowConfig

Represents a setup node for creating an embedded flow. This type extends the Metadata metadata type and inherits its `fullName` field.

EmbeddedServiceLiveAgent

Represents a setup node for creating an embedded chat deployment. This type extends the Metadata metadata type and inherits its `fullName` field.

EmbeddedServiceMenuSettings

Represents a setup node for creating a channel menu deployment. Channel menus list the ways in which customers can contact your business. This type extends the Metadata metadata type and inherits its `fullName` field.

EntitlementProcess

Represents the settings for an entitlement process.

EntitlementTemplate

Represents an entitlement template. Entitlement templates are predefined terms of customer support that you can quickly add to products. For example, you can create entitlement templates for Web or phone support so that users can easily add entitlements to products offered to customers.

EscalationRules

Represents case escalation rules to escalate cases automatically if they are not resolved within a certain period of time. You can access rules metadata for all applicable objects, for a specific object, or for a specific rule on a specific object.

EventDelivery

Represents how an event instance maps to a target payload. Removed in API version 46.0. This type extends the Metadata metadata type and inherits its `fullName` field.

EventSubscription

Represents a subscription to an event type. Removed in API version 46.0. This type extends the Metadata metadata type and inherits its `fullName` field.

ExperienceBundle

Represents a text-based code structure of community settings and site components, such as pages, branding sets, and themes. Developers can quickly update and deploy one or more Lightning communities *programmatically* using their preferred development tools.This type extends the Metadata metadata type and inherits its `fullName` field.

ExternalDataSource

Represents the metadata associated with an external data source. Create external data sources to manage connection details for integration with data and content that are stored outside your Salesforce org.

ExternalServiceRegistration

Represents the External Service configuration for an org. This type extends the Metadata metadata type and inherits its `fullName` field.

FeatureParameterBoolean

Represents a boolean feature parameter in a packaging org that has access to the Feature Management App (FMA). Feature parameters let you drive app behavior and track activation metrics in subscriber orgs that install your package. This type extends the Metadata metadata type and inherits its `fullName` field.

FeatureParameterDate

Represents a date feature parameter in a packaging org that has access to the Feature Management App (FMA). Feature parameters let you drive app behavior and track activation metrics in subscriber orgs that install your package. This type extends the Metadata metadata type and inherits its `fullName` field.

FeatureParameterInteger

Represents an integer feature parameter in a packaging org that has access to the Feature Management App (FMA). Feature parameters let you drive app behavior and track activation metrics in subscriber orgs that install your package. This type extends the Metadata metadata type and inherits its `fullName` field.

FlexiPage

Represents the metadata associated with a Lightning page. A Lightning page represents a customizable screen made up of regions containing Lightning components.

Flow

Represents the metadata associated with a flow. With Flow, you can create an application that navigates users through a series of screens to query and update records in the database. You can also execute logic and provide branching capability based on user input to build dynamic applications.

FlowCategory

Represents a list of flows that are grouped by category. Flows aren't added directly to a Lightning Bolt Solution. Instead, add the category the flows are in to the Lightning Bolt Solution. This type extends the Metadata metadata type and inherits its `fullName` field.

FlowDefinition

Represents the flow definition's description and active flow version number.

Folder

Represents a folder. This type extends the Metadata metadata type and inherits its `fullName` field.

GlobalPicklist

Represents a global picklist, or the set of shared picklist values that custom picklist fields can use. (In contrast, the custom picklist fields that are based on a global picklist are of type CustomValue.) This type extends the Metadata metadata type and inherits its `fullName` field.

GlobalPicklistValue

Represents the definition of a value used in a global picklist. Custom picklist fields can inherit the picklist value set from a global picklist. This type extends the Metadata metadata type and inherits its `fullName` field.

GlobalValueSet

Represents the metadata for a global picklist value set, which is the set of shared values that custom picklist fields can use. A global value set isn't a field itself. (In contrast, the custom picklist fields that are based on a global picklist are of type ValueSet.) This type extends the Metadata metadata type and inherits its `fullName` field.

GlobalValueSetTranslation

Contains details for a global value set translation. Global value sets are lists of values that can be shared by multiple custom picklist fields, optionally across objects.This type extends the Metadata metadata type and inherits its `fullName` field.

Group

Represents a set of public groups, which can have users, roles, and other groups.

HomePageComponent

Represents the metadata associated with a home page component. You can customize the Home tab in Salesforce Classic to include components such as sidebar links, a company logo, a dashboard snapshot, or custom components that you create. Use to create, update, or delete home page component definitions.

HomePageLayout

Represents the metadata associated with a home page layout. You can customize home page layouts and assign the layouts to users based on their user profile.

InstalledPackage

Represents a package to be installed or uninstalled. Deploying a newer version of a currently installed package upgrades the package.

KeywordList

Represents a list of keywords used in community moderation. This keyword list is a type of moderation criteria that defines offensive language or inappropriate content that you don’t want in your community.

Layout

Represents the metadata associated with a page layout. For more information, see “Page Layouts” in the Salesforce Help.

Letterhead

Represents formatting options for the letterhead in an email template. A letterhead defines the logo, page color, and text settings for your HTML email templates. Use letterheads to ensure a consistent look and feel in your company’s emails.

LightningBolt

Represents the definition of a Lightning Bolt Solution, which can include custom apps, flow categories, and Lightning Community templates. This type extends the Metadata metadata type and inherits its `fullName` field.

LightningComponentBundle

Represents a Lightning web component bundle. A bundle contains Lightning web component resources.

LightningExperienceTheme

Represents the details of a custom theme, including the BrandingSet. Themes enable admins to specify configurable attributes, such as three colors and five images. The colors and some of the images override SLDS token values and influence the generation of `app.css`.

LightningMessageChannel (Beta)

Represents the metadata associated with a Lightning Message Channel. A Lightning Message Channel represents a secure channel to communicate across UI technologies (Lightning Web Components, Aura Components, and Visualforce). This type extends the Metadata on page 652 metadata type and inherits its `fullName` field.

LiveChatAgentConfig

Represents the configuration of an organization’s Chat deployment, such as how many chats can be assigned to an agent and whether or not chat sounds are enabled.

LiveChatButton

Represents a Chat deployment’s settings for the button that customers click to chat with an agent and the chat window, such as the label that appears on the button and the pre-chat form that appears before a chat begins.

LiveChatDeployment

Represents the configuration settings for a specific Chat deployment, such as the branding image for the deployment and whether or not chat transcripts are automatically saved.

LiveChatSensitiveDataRule

Represents a rule for masking or deleting data of a specified pattern. Written as a regular expression (regex).

ManagedContentType

Represents the definition of custom content types for use with Salesforce CMS. Custom content types are displayed as forms with defined fields. This type extends the Metadata metadata type and inherits its `fullName` field.

ManagedTopics

Represents navigational and featured topics managed in a community. A specific community is represented by the Network component.

MatchingRule

Represents a matching rule that is used to identify duplicate records. This type extends the Metadata metadata type and inherits its `fullName` field.

Metadata

This is the base class for all metadata types. You cannot edit this object. A component is an instance of a metadata type.

MetadataWithContent

MetadataWithContent is the base type for all metadata types that contain content, such as documents or email templates. It extends Metadata. You cannot edit this object.

MilestoneType

Represents the name and description of a milestone, which you can use in an entitlement process to track important steps in cases.

MlDomain (Beta)

Represents an Einstein Intent Set. This type extends the Metadata metadata type and inherits its `fullName` field.

MobileApplicationDetail

Represents the packaging attributes for a mobile connected app. This type extends the Metadata metadata type and inherits its `fullName` field.

ModerationRule

Represents a rule used in your community to moderate member-generated content.Each rule specifies the member-generated content the rule applies to, the criteria to enforce the rule on, and the moderation action to take. Moderation rules help protect your community from spammers, bots, and offensive or inappropriate content. This type extends the Metadata metadata type and inherits its `fullName` field.

MyDomainDiscoverableLogin

Represents the configuration settings when the My Domain login page type is Discovery. Login Discovery provides an identity-first login experience, where the login page contains the identifier field only. Based on the identifier entered, a handler determines how to authenticate the user. This type extends the Metadata metadata type and inherits its `fullName` field.

NamedCredential

Represents a named credential, which specifies the URL of a callout endpoint and its required authentication parameters in one definition. A named credential can be specified as an endpoint to simplify the setup of authenticated callouts.

NavigationMenu

Represents the navigation menu in a community. A navigation menu consists of items that users can click to go to other parts of the community. This type replaces the NavigationLinkSet subtype on Network. NavigationMenu is available in API version 47.0 and later.This type extends the Metadata metadata type and inherits its `fullName` field.

Network

Represents a community. Communities are branded spaces for your employees, customers, and partners to connect. You can customize and create communities to meet your business needs, then transition seamlessly between them. Use the Network component for Salesforce Communities. If you want to create zones that contain Chatter Answers and Ideas, use the Community (Zone) component.

NetworkBranding

Represents the branding and color scheme applied to the login pages of a community. (Salesforce Communities are represented by the Network component.)

NotificationTypeConfig

Represents the metadata associated with org-level notification settings for standard and custom notification types. This type extends the Metadata metadata type and inherits its `fullName` field.

OauthCustomScope

Represents a permission defining the protected data that a connected app can access from an external entity when Salesforce is the OAuth authorization provider.

Package

Specifies which metadata components to retrieve as part of a `retrieve()` call or defines a package of components.

PathAssistant

Represents Path records.This type extends the Metadata metadata type and inherits its `fullName` field.

PaymentGatewayProvider

Represents the metadata associated with a payment gateway provider. This type extends the Metadata metadata type and inherits its `fullName` field.

PermissionSet

Represents a set of permissions that's used to grant additional access to one or more users without changing their profile or reassigning profiles. You can use permission sets to grant access but not to deny access.

PermissionSetGroup

Represents a group of permission *sets* and the permissions within them. Use permission set groups to organize permissions based on job functions or tasks. Then, you can package the groups as needed.

PlatformCachePartition

Represents a partition in the Platform Cache. This type extends the Metadata metadata type and inherits its `fullName` field.

PlatformEventChannel

Represents a Change Data Capture standard or custom channel that you can subscribe to for receiving notifications of Salesforce record changes. The standard channel corresponds to the entity selection in the Change Data Capture page in Setup. A custom channel is a channel that you define using this metadata type. Starting in API version 47.0, the channel doesn't contain the selected entities, which are represented each by PlatformEventChannelMember. In API version 46.0 and earlier, the channel includes the selected entities. Change Data Capture sends notifications for created, updated, deleted, and undeleted records only for the selected entities. This type extends the Metadata on page 652 metadata type and inherits its `fullName` field.

PlatformEventChannelMember

Represents an entity selected for Change Data Capture notifications on a standard or custom channel. This type extends the Metadata on page 652 metadata type and inherits its `fullName` field.

Portal

The Portal metadata type represents a partner portal or Customer Portal.

PostTemplate

Represents the metadata associated with an approval post template for Approvals in Chatter. With approval post templates, you can customize the information included in approval request posts that appear in Chatter feeds. This type extends the Metadata metadata type and inherits its `fullName` field.

PresenceDeclineReason

Represents an Omni-Channel decline reason that agents can select when declining work requests. This type extends the Metadata metadata type and inherits its `fullName` field.

PresenceUserConfig

Represents a configuration that determines a presence user's settings. This type extends the Metadata metadata type and inherits its `fullName` field.

Profile

Represents a user profile. A profile defines a user's permission to perform different functions within Salesforce. This type extends the Metadata metadata type and inherits its `fullName` field.

ProfileActionOverride

Represents an override of an ActionOverride by a user profile. You can use it to override an ActionOverride on a standard Home tab or object record page in Lightning Experience. When a user logs in with a profile, a matching ProfileActionOverride assignment takes precedence over existing overrides for the Home tab or record page specified in ActionOverride. In API versions 39.0 to 44.0, you can access ProfileActionOverride by accessing its encompassing CustomApplication or Profile metadata types. In API version 45.0 and later, you can access ProfileActionOverride only by accessing its encompassing CustomApplication.

ProfilePasswordPolicy

Represents a profile's password policies. Profile password policies override org-wide password policies for that profile's users. Use ProfilePasswordPolicy to retrieve password policies for a given profile. This type extends the Metadata metadata type and inherits its `fullName` field.

ProfileSessionSetting

Represents a profile's session settings. Use ProfileSessionSetting to retrieve the session settings for a given profile. This type extends the Metadata metadata type and inherits its `fullName` field.

Prompt

Represents the metadata associated with in-app guidance prompts. Use prompts to display announcements, training, or news to users within the app. Choose to add action button that links to a URL of your choice. Track view and button click completes.This type extends the Metadata metadata type and inherits its `fullName` field.

Queue

Represents a holding area for items before they are processed.

QueueRoutingConfig

Represents the settings that determine how work items are routed to agents.This type extends the Metadata metadata type and inherits its `fullName` field.

QuickAction

Represents a specified create or update quick action for an object that then becomes available in the Chatter publisher. For example, you can create an action that, on the detail page of an account, allows a user to create a contact related to that account from the Chatter feed on that page. QuickAction can be created on objects that allow custom fields.

RedirectWhitelistUrl

Represents a whitelisted URL that users can navigate to without being shown a warning message. This type extends the Metadata metadata type and inherits its `fullName` field.

RecommendationStrategy

Represents a recommendation strategy. Recommendation strategies are applications, similar to data flows, that determine a set of recommendations to be delivered to the client through data retrieval, branching, and logic operations.

RecordActionDeployment

Represents configuration settings for the Actions & Recommendations component. For example, you can have a deployment that specifies which types of actions to display, default actions for channels, and the actions that users can add at runtime. If the component shows Next Best Action recommendations, the deployment configures which strategies to use and how recommendations appear. This type extends the Metadata metadata type and inherits its `fullName` field.

RemoteSiteSetting

Represents a remote site setting. Before any Visualforce page, Apex callout, or JavaScript code using XmlHttpRequest in an s-control or custom button can call an external site, that site must be registered in the Remote Site Settings page, or the call fails.

Report

Represents a custom report. This metadata type only supports custom reports; standard reports are not supported.

ReportType

Represents the metadata associated with a custom report type. Custom report types allow you to build a framework from which users can create and customize reports. For more information, see "Set Up a Custom Report Type" in the Salesforce online help.

Role

Represents a role in your organization.

RoleOrTerritory

Represents the common base type and valid values for role or territory.

SamlSsoConfig

Represents a SAML Single Sign-On configuration. This type extends the Metadata metadata type and inherits its `fullName` field. Single sign-on (SSO) lets users access authorized network resources with one login. You validate usernames and passwords against your corporate user database or other client app rather than Salesforce managing separate passwords for each resource.

Scontrol

Deprecated. Represents an Scontrol component, corresponding to an s-control in the Salesforce user interface.

ServiceChannel

Represents a channel of work items that are received from your organization—for example, cases, chats, or leads. This type extends the Metadata metadata type and inherits its `fullName` field.

ServicePresenceStatus

Represents a presence status that can be assigned to a service channel. This type extends the Metadata metadata type and inherits its `fullName` field.

Settings

Represents the organization settings related to a feature. For example, your password policies, session settings and network access controls are all available in the SecuritySettings component type.

SharedTo

SharedTo defines the sharing access for a list view or a folder. It can be used to specify the target and source for owner-based sharing rules.

SharingBaseRule

Represents sharing rule settings such as access level and to whom access is granted.

SharingRules

Represents the base container for sharing rules, which can be criteria-based, ownership-based, territory-based, or for guest user access. SharingRules enables you to share records with a set of users, using rules that specify the access level for the target user group.

SharingSet

Represents a sharing set. A sharing set defines an access mapping that grants portal or community users access to objects that are associated with their accounts or contacts. This type extends the Metadata metadata type and inherits its `fullName` field.

SiteDotCom

Represents a site for deployment.

Skill

Represents the settings for a skill used for field service or to route chats to agents in Chat, such as the name of the skill and which agents the skills are assigned to.

StandardValueSet

Represents the set of values in a standard picklist field. This type extends the Metadata metadata type and inherits its `fullName` field.

StandardValueSetTranslation

Contains details for a standard picklist translation. It returns a translated standard value set.This type extends the Metadata metadata type and inherits its `fullName` field.

StaticResource

Represents a static resource file, often a code library in a ZIP file. Static resources allow you to upload content that you can reference in a Visualforce page, including archives (such as .zip and .jar files), images, style sheets, JavaScript, and other files.

SynonymDictionary

Represents a set of synonym groups, which are groups of words or phrases that are treated as equivalent in users' searches. You can define synonym groups to optimize search results for acronyms, variations of product names, and other terminology unique to your organization.

Territory

Represents a territory in your organization.

Territory2

Represents the metadata associated with a sales territory in Territory Management 2.0. This type extends the Metadata metadata type and inherits its `fullName` field.Available only if Territory Management 2.0 has been enabled for your organization.

Territory2Model

Represents the metadata associated with a territory model in Territory Management 2.0. This type extends the Metadata metadata type and inherits its `fullName` field.Available only if Territory Management 2.0 has been enabled for your Salesforce org.

Territory2Rule

Represents the metadata associated with a territory assignment rule associated with an object, such as Account, in Territory Management 2.0. This type extends the Metadata metadata type and inherits its `fullName` field.Available only if Territory Management 2.0 has been enabled for your Salesforce org.

Territory2Type

Represents the metadata for a category of territories in Territory Management 2.0. Every Territory2 must have a Territory2Type. This type extends the Metadata metadata type and inherits its `fullName` field.Available only if Enterprise Territory Management has been enabled for your Salesforce org.

TimeSheetTemplate

Represents a template for creating time sheets in Field Service Lightning.This type extends the Metadata metadata type and inherits its `fullName` field.

TopicsForObjects

Represents the ability to assign topics to objects or to remove topic assignments.

TransactionSecurityPolicy

Represents a transaction security policy definition. Transaction Security policies give you a way to look through events in your organization and specify actions to take when certain combinations occur.

Translations

This metadata type allows you to work with translations for various supported languages. The ability to translate component labels is part of the Translation Workbench. For more information, see "Enable and Disable the Translation Workbench" in the Salesforce online help.

UserCriteria

Represents the member criteria to use in community moderation rules. This type extends the Metadata metadata type and inherits its `fullName` field..

WaveApplication

Represents the Analytics application. This type extends the Metadata metadata type and inherits its `fullName` field.

WaveDataflow

Represents the WaveDataflow object in the Analytics application. This type extends the MetadataWithContent metadata type and inherits its `content` and `fullName` fields.

WaveDashboard

Represents the WaveDashboard object in the Analytics application. This type extends the MetadataWithContent metadata type and inherits its `content` and `fullName` fields.

WaveDataset

Represents the WaveDataset object in the Analytics application. This type extends the Metadata metadata type and inherits its `fullName` field.

WaveLens

Represents the WaveLens object in the Analytics application. This type extends the MetadataWithContent metadata type and inherits its `content` and `fullName` fields.

WaveRecipe

Represents the WaveRecipe object in the Analytics application. This type extends the MetadataWithContent metadata type and inherits its `content` and `fullName` fields.

WaveTemplateBundle

Represents an Analytics template bundle, which can be used to create Analytics apps. A bundle contains an Analytics template definition and all its related resources.This type extends the Metadata metadata type and inherits its `fullName` field.

WaveXmd

Represents the WaveXmd object in the Analytics application. This type extends the Metadata metadata type and inherits its `fullName` field.

Workflow

Represents the metadata associated with a workflow rule. A workflow rule sets workflow actions into motion when its designated conditions are met. You can configure workflow actions to execute immediately when a record meets the conditions in your workflow rule, or set time triggers that execute the workflow actions on a specific day. Use this metadata type to create, update, or delete workflow rule definitions.

WorkSkillRouting

Represents a setup object that stores a set of WorkSkillRoutingAttribute objects. These objects are used to route a work item to an agent who has the skills necessary to take the work. This type extends the Metadata metadata type and inherits its `fullName` field.

Metadata Components and Types

Metadata components are not based on sObjects, like objects in the API. Instead, they are based on metadata types, such as ApexClass and CustomObject, which extend Metadata, the base class for all metadata types. A component is an instance of a metadata type.

For example, `CustomObject` is a metadata type for custom objects, and the `MyCustomObject__c` component is an instance of a custom object.

A metadata type can be identified in the metadata WSDL as any complexType that extends the Metadata complexType. A complexType that is a metadata type includes the following element in its WSDL definition:

```
<xsd:extension base="tns:Metadata">
```

CustomObject and BusinessProcess extend Metadata so they are metadata types; ActionOverride doesn't extend Metadata so it's not a metadata type.

You can individually deploy or retrieve a component for a metadata type. For example, you can retrieve an individual BusinessProcess component, but you can't retrieve an individual ActionOverride component. You can only retrieve an ActionOverride component by retrieving its encompassing CustomObject component.

Metadata components can be manipulated by asynchronous Metadata API calls or declarative (or file-based) Metadata API calls.

Most of the components can be accessed using Salesforce Extensions for Visual Studio Code. Exceptions are noted in the description of the object.

Field Data Types

Each component field has a specific field type. These field types can correspond to other components defined in the WSDL, or primitive data types, like `string`, that are commonly used in strongly typed programming languages.

These field data types are used in the SOAP messages that are exchanged between your client application and the API. When writing your client application, follow the data typing rules defined for your programming language and development environment. Your development tool handles the mapping of typed data in your programming language with these SOAP data types.

For more information about primitive data types, see the *SOAP API Developer Guide*.

Enumeration Fields

Some component fields have a data type that is an enumeration. An enumeration is the API equivalent of a picklist. The valid values of the field are restricted to a strict set of possible values, all having the same data type. These values are listed in the field description column for each enumeration field. See `sortBy` for an example of an enumeration field of type string. The XML below shows a sample definition of an enumeration of type string in the WSDL.

```
<xsd:simpleType name="DashboardComponentFilter">
    <xsd:restriction base="xsd:string">
        <xsd:enumeration value="RowLabelAscending"/>
        <xsd:enumeration value="RowLabelDescending"/>
        <xsd:enumeration value="RowValueAscending"/>
        <xsd:enumeration value="RowValueDescending"/>
    </xsd:restriction>
</xsd:simpleType>
```

Supported Calls

All of the metadata types are supported by the main calls, unless it is stated otherwise in the individual component sections. The main Metadata API calls are:

- CRUD calls, such as createMetadata() and deleteMetadata()
- File-based calls, such as deploy() and retrieve()
- Utility calls, such as listMetadata() and describeMetadata()

Metadata Coverage Report

Launch the Metadata Coverage report to determine supported metadata components. The Metadata Coverage report is the ultimate source of truth for metadata coverage across several channels. These channels include Metadata API, scratch org source tracking, unlocked packages, second-generation managed packages, classic managed packages, and more.

To view the Metadata Coverage report, you don't have to be logged into an org.

Unsupported Metadata Types

Some things you can customize in a Salesforce org aren't available in Metadata API.

The following components can't be retrieved or deployed with Metadata API, and changes to them must be made manually in each of your organizations:

- Account Teams
- Activity Button Overrides
- Auto-number on Customizable Standard Fields
- Calendars
- Campaign Influences
- Case Contact Roles
- Case Feed Layouts
- Case Team Roles
- Console Layouts
- Multiline layout fields for contract line items
- Currency Exchange Rates
- Data Category Visibility Settings
- Delegated Administration
- Divisions
- File Upload and Download Security Settings
- Mail Merge Templates
- Multiline layout fields for opportunity teams
- Offline Briefcase Configurations
- Omni-Channel Queues and Omni-Channel Skills routing types for the LiveChatButton object
- Opportunity Big Deal Alerts
- Opportunity Update Reminders

- Organization Wide Email Addresses
- Partner Management
- The following standard picklists: IdeaTheme.Categories, Question.Origin. (All other standard picklists are supported.)
- Predefined Case Teams
- Quote Templates
- Salesforce to Salesforce
- Self-Service Portal Font and Colors
- Self-Service Portal Users
- Self-Service Public Solutions
- Self-Service Web-to-Case
- Service report templates
- Social Business Rules
- SoftPhone Layout
- Solution Categories
- Solution Settings
- Standard fields that aren't customizable, such as autonumber fields or system fields
- Web Links on Person Account Page Layouts
- Web-to-Lead

Special Behavior in Metadata API Deployments

Important considerations for specific types and contents of a deployment.

When deploying changes to a Salesforce org, consider how individual components in your deployment behave so you're including all the necessary changes. Use the following information to determine what to include in your deployment, and how the changes appear in the destination org.

Apex Classes and Apex Triggers

By default, changes to Apex code that has Apex jobs pending or in progress can't be deployed. To deploy these changes, do one of the following.

- Cancel Apex jobs before deploying changes to Apex code. Reschedule the jobs after the deployment.
- Enable deployments with Apex jobs in the Salesforce user interface in the Deployment Settings page.

Approval Processes

- To use approval processes on Salesforce Knowledge articles with the Metadata API, the article type must be deployed. For article version (_kav) in approval processes, the supported action types are: Knowledge Action, Email Alert, Field Update, and Outbound Message.
- If the approval process references any post templates that contain custom fields, then you need to resave those post templates in the originating organization before adding them to the change set. From Setup, enter *`Post Templates`* in the `Quick Find` box, then select **Post Templates**. For each post template, click **Edit** and then **Save**.
- The metadata doesn't include the order of active approval processes. You might need to reorder the approval processes in the destination org after deployment.
- If you change the `Unique Name` of an approval process that was previously included in a change set and deployed in another organization, and you resend the approval process via a change set, a new approval process will be created upon deployment in the other organization. The previously deployed approval process will not be modified.

Custom Fields

Starting in API version 30.0, when deploying a new custom field, the default values for the `editable` and `readable` fields in profile field permissions are `false`. To override the default values, include field permissions for the new field in your profiles.

Custom Objects

Using API version 29.0, you can't change the `sharingModel` of an object using Metadata API. You must manually make this change to the target org through the user interface.

Starting with API version 30.0, you can change the `sharingModel` of an object for internal users using Metadata API and the user interface.

Connected App

- You cannot set the `consumerKey` in Metadata API. It is included in a retrieve operation for informational purposes. If you try to move the connected app to another org, you must remove the `consumerKey` from the .zip file before the deployment to an org. A new key will be generated in the destination org.
- Mobile settings of connected apps are not supported in change sets and must be manually migrated.

Groups

Members of the public group are not migrated when you deploy the group type.

Master-Detail Relationships

A Metadata API deployment that includes Master-Detail relationships deletes all detail records in the Recycle Bin in the following cases.

1. For a deployment with a new Master-Detail field, soft delete (send to the Recycle Bin) all detail records before proceeding to deploy the Master-Detail field, or the deployment fails. During the deployment, detail records are permanently deleted from the Recycle Bin and cannot be recovered.
2. For a deployment that converts a Lookup field relationship to a Master-Detail relationship, detail records must reference a master record or be soft-deleted (sent to the Recycle Bin) for the deployment to succeed. However, a successful deployment permanently deletes any detail records in the Recycle Bin.

Page Layout

A deployment containing page layout assignments replaces all existing page layout assignments in the destination org with those specified in the .zip file. Existing page layouts in the org disappear if they're not included in the .zip file. Always include all page layouts for all required record types in the .zip file.

Picklist Values

Values for a picklist field in a target org that are not included in the metadata are set to inactive.

For example, if the target org has a picklist that includes an active value of `1`, and the metadata does not include `1` as a value for the picklist, `1` changes from active to inactive in the target org.

Profiles

If a package includes a profile with a name that doesn't exist in the target org, a new profile is created with that name. If the deployed profile doesn't specify any permissions or settings, the resulting profile consists of all the permissions and settings in the Standard Profile.

Sharing

- Simultaneously updating the `sharingModel` field for an object and adding a new sharing rule isn't supported in Metadata API, regardless of which object you're updating. For example, you can add a sharing rule when the org-wide default is public, and subsequently update the `sharingModel`. This would result in a single sharing recalculation.
- You might encounter an error if you're deploying a change set with a custom object that has a parent-child relationship without the master/detail field in the same change set. To resolve this error, include the master/detail custom field in the change set, even if you haven't changed the org-wide default.

Workflow

Test mode for flow triggers isn't supported in the Metadata API. If you want a flow trigger to run the latest flow version when an administrator causes the workflow rule to fire, enable test mode via the user interface after deployment.

AccountRelationshipShareRule

Represents the rule that determines which object records are shared, how they are shared, the account relationship type that shares the records, and the level of access granted to the records. This type extends the MetadataWithContent metadata type and inherits its `content` and `fullName` fields.

File Suffix and Directory Location

AccountRelationshipShareRule components have the suffix `.accountRelationshipShareRule` and are stored in the `.accountRelationshipShareRules` folder.

Version

AccountRelationshipShareRule components are available in API version and later.

Special Access Rules

Access to the Accountrelationshipsharerule type requires orgs to enable the Account Relationships permission.The Manage Communities permission is required for user access.

Fields

Field Name	Field Type	Description
`accessLevel`	string	Type of access granted by the share rule. Valid values are: • Read • Edit
`accountToCriteriaField`	string	Criteria that must be met for the data to be shared. Valid values are: • Account.OwnerId • Account.ParentId • Campaign.OwnerId • Case.AccountId • Case.OwnerId • Contact.AccountId • Contact.OwnerId • Lead.ConvertedAccountId • Lead.OwnerId • Lead.PartnerAccountId

Field Name	Field Type	Description
		- Opportunity.AccountId - Opportunity.OwnerId - Opportunity.PartnerAccountId - Order.AccountId - Order.OwnerId To get the full list for your org, do a Describe on the ARSR entity.
`description`	string	A meaningful explanation of the sharing rule.
`entityType`	string	The type of data being shared by this share rule. Valid values are: - Account - Campaign - Case - Contact - Lead - Opportunity - Order API names of top level custom objects in the org may also be used, e.g. CustomObject__c.
`masterLabel`	string	The label assigned to the sharing rule to identify it.
`staticFormulaCriteria`	string	A way to further filter what data gets shared. This must be a deterministic formula and spanning is not allowed.
`type`	string	Must match the type of an account relationship for data to be shared according to the accountToCriteriaField and the staticFormulaCriteria fields.

Declarative Metadata Sample Definition

The following is an example of an AccountRelationshipShareRule component.

```
<?xml version="1.0" encoding="UTF-8"?>
<AccountRelationshipShareRule xmlns="http://soap.sforce.com/2006/04/metadata">
    <accessLevel>Edit</accessLevel>
    <accountToCriteriaField>Account.OwnerId</accountToCriteriaField>
    <description>TestDescription</description>
    <entityType>Account</entityType>
    <masterLabel>TestName</masterLabel>
    <staticFormulaCriteria>YearStarted = &quot;1980&quot;</staticFormulaCriteria>
    <type>Dealer</type>
</AccountRelationshipShareRule>
```

The following is an example `package.xml` that references the previous definition.

```
<Package>
<?xml version="1.0" encoding="UTF-8"?>
<Package xmlns="http://soap.sforce.com/2006/04/metadata">
    <types>
        <members>ArsrDevName</members>
        <name>AccountRelationshipShareRule</name>
    </types>
<version>45.0</version>
</Package>
```

This metadata type supports the wildcard character * (asterisk) in the package.xml manifest file.

ActionLinkGroupTemplate

Represents the action link group template. Action link templates let you reuse action link definitions and package and distribute action links. An action link is a button on a feed element. Clicking on an action link can take a user to another Web page, initiate a file download, or invoke an API call to an external server or Salesforce. Use action links to integrate Salesforce and third-party services into the feed. Every action link belongs to an action link group and action links within the group are mutually exclusive.

This type extends the Metadata metadata type and inherits its `fullName` field.

File Suffix and Directory Location

ActionLinkGroupTemplate components have the suffix `.actionLinkGroupTemplate` and are stored in the `actionLinkGroupTemplates` folder.

Version

ActionLinkGroupTemplate components are available in API version 33.0 and later.

Fields

Field Name	Field Type	Description
`actionLinkTemplates`	ActionLinkTemplate on page 142[]	Action link templates that are associated with the action link group template.
`category`	PlatformAction GroupCategory (enumeration of type string)	Required. The location of the action link group within the feed element. Values are: • `Primary`—The action link group is displayed in the body of the feed element. • `Overflow`—The action link group is displayed in the overflow menu of the feed element.

Field Name	Field Type	Description
executionsAllowed	ActionLink ExecutionsAllowed (enumeration of type string)	Required. The number of times an action link can be executed. Values are: • Once—An action link can be executed only once across all users. • OncePerUser—An action link can be executed only once for each user. • Unlimited—An action link can be executed an unlimited number of times by each user. If the action link's actionType is Api or ApiAsync, you can't use this value.
hoursUntilExpiration	int	Required. The number of hours from when the action link group is created until it's removed from associated feed elements and can no longer be executed. The maximum value is 8,760.
isPublished	boolean	Required. If true, the action link group template is published. Action link group templates shouldn't be published until at least one action link template is associated with it.
name	string	Required. The name of the action link group template to use in code.

ActionLinkTemplate

ActionLinkTemplate components are used to create multiple action links that share properties.

Field Name	Field Type	Description
actionUrl	string	Required. The action link URL. For example, a Ui action link URL is a Web page. A Download action link URL is a link to the file to download. Ui and Download action link URLs are provided to clients. An Api or ApiAsync action link URL is a REST resource. Api and ApiAsync action link URLs aren't provided to clients. Links to Salesforce can be relative. All other links must be absolute and start with https://.
headers	string	Template for the HTTP headers sent when corresponding action links are invoked. This field can be used only for Api and ApiAsync action links. This field can contain context variables and binding variables in the form {!Bindings.***key***}.
isConfirmationRequired	boolean	Required. If true, a confirmation dialog appears before the action is executed.
isGroupDefault	boolean	Required. If true, action links derived from this template are the default or primary action in their action groups. There can be only one default action per action group.
label	string	A custom label to display on the action link button. If none of the LabelKey values make sense for an action link, use a custom label. Set the LabelKey field to None and enter a label name in the Label field.
labelKey	string	Required. Key for the set of labels to display for these action link states: new, pending, success, failed. For example, the Approve set contains these labels:

Field Name	Field Type	Description
		Approve, Pending, Approved, Failed. For a complete list of keys and labels, see Action Links Labels in the *Chatter REST API Developer Guide* or the *Apex Developer Guide*.
`linkType`	ActionLinkType (enumeration of type string)	Required. The type of action link. One of these values: • `Api`—The action link calls a synchronous API at the action URL. Salesforce sets the status to `SuccessfulStatus` or `FailedStatus` based on the HTTP status code returned by your server. • `ApiAsync`—The action link calls an asynchronous API at the action URL. The action remains in a `PendingStatus` state until a third party makes a request to `/connect/action-links/`***`actionLinkId`*** to set the status to `SuccessfulStatus` or `FailedStatus` when the asynchronous operation is complete. • `Download`—The action link downloads a file from the action URL. • `Ui`—The action link takes the user to a web page at the action URL.
`method`	ActionLink HttpMethod (enumeration of type string)	Required. HTTP method for the action URL. One of these values: • `HttpDelete`—Returns HTTP 204 on success. Response body or output class is empty. • `HttpGet`—Returns HTTP 200 on success. • `HttpHead`—Returns HTTP 200 on success. Response body or output class is empty. • `HttpPatch`—Returns HTTP 200 on success or HTTP 204 if the response body or output class is empty. • `HttpPost`—Returns HTTP 201 on success or HTTP 204 if the response body or output class is empty. Exceptions are the batch posting resources and methods, which return HTTP 200 on success. • `HttpPut`—Return HTTP 200 on success or HTTP 204 if the response body or output class is empty. `Ui` and `Download` action links must use `HttpGet`.
`position`	int	Required. An integer specifying the position of the action link template relative to other action links in the group. 0 is the first position.
`requestBody`	string	Template for the HTTP request body sent when corresponding action links are invoked. This field can be used only for `Api` and `ApiAsync` action links. This field can contain context variables and binding variables in the form `{!Bindings.`***`key`***`}`.
`userAlias`	string	If you selected `CustomUser` or `CustomExcludedUser` for `UserVisibility`, this field is the alias for the custom user. Use the alias in a template binding to specify the custom user when an action link group is created using the template.

Field Name	Field Type	Description
`userVisibility`	ActionLink UserVisibility (enumeration of type string)	Required. Who can see the action link. This value is set per action link, not per action link group. Values are: • `Creator`—Only the creator of the action link can see the action link. • `Everyone`—Everyone can see the action link. • `EveryoneButCreator`—Everyone but the creator of the action link can see the action link. • `Manager`—Only the manager of the creator of the action link can see the action link. • `CustomUser`—Only the custom user can see the action link. • `CustomExcludedUser`—Everyone but the custom user can see the action link.

Declarative Metadata Sample Definition

The following is an example of an ActionLinkGroupTemplate component.

```
<?xml version="1.0" encoding="UTF-8"?>
<ActionLinkGroupTemplate xmlns="http://soap.sforce.com/2006/04/metadata">
   <actionLinkTemplates>
      <actionUrl>/services/data/{!Bindings.word}/chatter/feed-elements</actionUrl>
      <headers>Content-Type:{!Bindings.word3}</headers>
      <isConfirmationRequired>true</isConfirmationRequired>
      <isGroupDefault>true</isGroupDefault>
      <labelKey>Add</labelKey>
      <linkType>API</linkType>
      <method>httpPost</method>
      <position>0</position>
      <requestBody>{"body":{"messageSegments":[{"type": "Text",
      "text": "{!Bindings.word1}"}]},"subjectId": "{!Bindings.word2}",
      "feedElementType": "feedItem"}</requestBody>
      <userAlias>customExcludedUser</userAlias>
      <userVisibility>CustomExcludedUser</userVisibility>
   </actionLinkTemplates>
   <category>Primary</category>
   <executionsAllowed>OncePerUser</executionsAllowed>
   <hoursUntilExpiration>10</hoursUntilExpiration>
   <isPublished>true</isPublished>
   <name>MyPackage</name>
</ActionLinkGroupTemplate>
```

The following is an example `package.xml` that references the previous definition.

```
<?xml version="1.0" encoding="UTF-8"?>
<Package xmlns="http://soap.sforce.com/2006/04/metadata">
    <types>
        <members>*</members>
        <name>ActionLinkGroupTemplate</name>
    </types>
    <version>33.0</version>
</Package>
```

Usage

If you modify action link group templates, you overwrite the related action link templates.

If you delete a published action link group template, you delete all related action link information which includes deleting all action links that were instantiated using the template from feed items.

Wildcard Support in the Manifest File

This metadata type supports the wildcard character `*` (asterisk) in the `package.xml` manifest file. For information about using the manifest file, see Deploying and Retrieving Metadata with the Zip File.

ActionPlanTemplate

Represents the instance of an action plan template.This type extends the Metadata metadata type and inherits its `fullName` field.

File Suffix and Directory Location

ActionPlanTemplate components have the suffix `.apt` and are stored in the `actionPlanTemplates` folder.

Version

Action Plan Template components are available in API version 47.0 and later.

Special Access Rules

To create or access action plan templates, you must have the Customize Application permission and the IndustriesActionPlans license.

Fields

Field Name	Field Type	Description
`actionPlanTemplateItem`	ActionPlanTemplateItem on page 146	The instance of an item on an action plan template version.
`description`	string	The description of the action plan template.
`name`	string	Required. The name of the action plan template.
`targetEntityType`	string	Required. The parent object this action plan template relates to. Supported parent objects are Account, BusinessMilestone, Campaign, Case, Claim, Contact, Contract, InsurancePolicy, InsurancePolicyCoverage, Lead, Opportunity, PersonLifeEvent, and Visit and custom objects with activities enabled.

Field Name	Field Type	Description
uniqueName	string	Required. The unique identifier for this action plan template record.

ActionPlanTemplateItem

Represents the instance of an item on an action plan template version.

Field Name	Field Type	Description
actionPlanTemplateItemValue	actionPlanTemplateItemValue	The value associated with the action plan template item.
displayOrder	int	The order in which this item is displayed within the action plan template version.
isRequired	boolean	Indicates whether the task created from this template item is required.
itemEntityType	string	Required. The name of the field on the action plan template item that this value is for. Available in API version 48.0 and later.
name	string	Required. The name of the action plan template item.
uniqueName	string	Required. The unique identifier for this action plan template item record.

ActionPlanTemplateItemValue

Represents the value associated with an action plan template item.

Field Name	Field Type	Description
itemEntityType	string	Required. The name of the field on the action plan template item that this value is for. Available in API version 48.0 and later.
name	string	Required. The name of the action plan template item value.
valueFormula	string	The formula for this action plan template item.
valueLiteral	string	The value for this action plan template item.

Declarative Metadata Sample Definition

The following is an example of an ActionPlanTemplate component.

```
<?xml version="1.0" encoding="UTF-8"?>
<ActionPlanTemplate xmlns="http://soap.sforce.com/2006/04/metadata">
    <actionPlanTemplateItem>
        <actionPlanTemplateItemValue>
            <itemEntityType>Task</itemEntityType>
            <name>Subject</name>
            <valueLiteral>Task 1</valueLiteral>
        </actionPlanTemplateItemValue>
        <actionPlanTemplateItemValue>
            <itemEntityType>Task</itemEntityType>
            <name>Priority</name>
            <valueLiteral>Normal</valueLiteral>
        </actionPlanTemplateItemValue>
        <actionPlanTemplateItemValue>
            <itemEntityType>Task</itemEntityType>
            <name>ActivityDate</name>
            <valueFormula>StartDate + 1</valueFormula>
        </actionPlanTemplateItemValue>
        <isRequired>true</isRequired>
        <itemEntityType>Task</itemEntityType>
        <name>Task 1</name>
        <uniqueName>Task_1_f1beaba5_1ae1_11ea_93ad_5bc214d4daaf</uniqueName>
    </actionPlanTemplateItem>
    <description>This is a sample template for packaging</description>
    <name>Action Plan Template 1</name>
    <targetEntityType>Account</targetEntityType>
    <uniqueName>Action_Plan_Template_1_da365953_1ae1_11ea_93ad_89a1c3d51bae</uniqueName>
</ActionPlanTemplate>
```

Wildcard Support in the Manifest File

This metadata type supports the wildcard character `*` (asterisk) in the `package.xml` manifest file. For information about using the manifest file, see Deploying and Retrieving Metadata with the Zip File.

AnalyticSnapshot

Represents a reporting snapshot. A reporting snapshot lets you report on historical data. Authorized users can save tabular or summary report results to fields on a custom object, then map those fields to corresponding fields on a target object. They can then schedule when to run the report to load the custom object's fields with the report's data. Reporting snapshots enable you to work with report data similarly to how you work with other records in Salesforce.

Declarative Metadata File Suffix and Directory Location

Lightning Platform AnalyticSnapshot components are stored in the `analyticSnapshots` directory of the corresponding package directory. The file name matches the unique name of the reporting snapshot, and the extension is `.analyticsnapshot`.

Version

Lightning Platform AnalyticSnapshot components are available in API version 16.0 and later.

Fields

Field	Field Type	Description
`description`	string	A description of the reporting snapshot.
`fullName`	string	The reporting snapshot name used for API access. The name can only contain characters, letters, and the underscore (_)

Field	Field Type	Description
		character, must start with a letter, and cannot end with an underscore or contain two consecutive underscore characters. This field is inherited from the Metadata component.
`groupColumn`	string	A column that specifies which level to extract data from the source report. It is only applicable for summary reports.
`mappings`	AnalyticSnapshotMapping[]	A list of reporting snapshot mappings. For valid values, see AnalyticSnapshotMapping.
`name`	string	Required. The display name of the reporting snapshot.
`runningUser`	string	The username of the user whose role and *sharing* settings are used to run the reporting snapshot.
`sourceReport`	string	Required. The report where data will be extracted from.
`targetObject`	string	Required. The custom object where data will be inserted into.

AnalyticSnapshotMapping

AnalyticSnapshotMapping defines the mapping for the reporting snapshot. Valid values are:

Field	Field Type	Description
`aggregateType`	ReportSummaryType[] (enumeration of type string)	List that defines if and how each report field is summarized. For valid values, see ReportSummaryType.
`sourceField`	string	The sourceField can be one of the following: • The field on the sourceReport that you want to map to the targetField in the targetObject • A summary of a filed on the sourceReport (for Summary reports only) • A field on the reporting snapshot, such as JobName, RunningUser, or ExecutionTime (set through the user interface) **Note:** The sourceField must correspond to the sourceType you specify.
`sourceType`	ReportJobSourceTypes[] (enumeration of type string)	List that defines the report format for the reporting snapshot. For valid values, see ReportJobSourceTypes.
`targetField`	string	A field on the targetObject into which this particular sourceField will be inserted.

ReportJobSourceTypes

An enumeration of type string that defines the report format for the reporting snapshot. Valid values are:

Enumeration Value	Description
`snapshot`	Use this option if the sourceField contains snapshot-specific information such as JobName, RunningUser, or ExecutionTime.
`summary`	Use this option if referencing a summary (Sum, Average, Minimum, Maximum) of a field from the sourceReport.
`tabular`	Use this option if referencing an available column from the sourceReport.

Declarative Metadata Sample Definition

Here's a sample XML definition of a reporting snapshot.

```xml
<?xml version="1.0" encoding="UTF-8"?>
<AnalyticSnapshot xmlns="http://soap.sforce.com/2006/04/metadata">
    <description>my description</description>
    <groupColumn>INDUSTRY</groupColumn>
    <mappings>
        <aggregateType>Average</aggregateType>
        <sourceField>SALES</sourceField>
        <sourceType>summary</sourceType>
        <targetField> myObject __c.Name</targetField>
    </mappings>
    <mappings>
        <sourceField>ExecutionTime</sourceField>
        <sourceType>snapshot</sourceType>
        <targetField> myObject __c.field3__c</targetField>
    </mappings>
    <mappings>
        <sourceField>INDUSTRY</sourceField>
        <sourceType>tabular</sourceType>
        <targetField>testObject__c.Name</targetField>
    </mappings>
    <name>my snapshot</name >
    <runningUser>user@salesforce.com</runningUser>
    <sourceReport>myFolder/mytSummaryReport</sourceReport>
    <targetObject>myObject__c</targetObject>
</AnalyticSnapshot>
```

Wildcard Support in the Manifest File

This metadata type doesn't support the wildcard character `*` (asterisk) in the `package.xml` manifest file. For information about using the manifest file, see Deploying and Retrieving Metadata with the Zip File.

SEE ALSO:

- Report

AnimationRule

Represents criteria for determining when an animation is displayed to Path users.This type extends the Metadata metadata type and inherits its `fullName` field.

File Suffix and Directory Location

AnimationRule components have the suffix `animationRule` and are stored in the `animationRules` folder.

Version

AnimationRule components are available in API version 46.0 and later.

Fields

Field Name	Field Type	Description
`animationFrequency`	picklist	Required. The frequency with which an animation is displayed when a user selects the designated picklist values in a path. Valid values are: • `always` • `often` • `sometimes` • `rarely` A value of `always` triggers an animation every time. The values `often`, `sometimes`, and `rarely` trigger an animation progressively less frequently.
`developerName`	string	Required. The developer name for the animation rule.
`isActive`	boolean	Required. Indicates whether the animation rule is active (`true`) or not (`false`).
`masterLabel`	string	Required. The label for the animation rule.
`recordTypeContext`	picklist	Required. An enum to track whether this AnimationRule applies to all record types for the associated sObject, or only to a single or master record type. Valid values are `All`, `Master`, or `Custom`.
`recordTypeName`	reference	The record type selected for the sObject in which the animation is displayed.
`sobjectType`	string	The object on which the animation rule is run.
`targetField`	string	Required. Name of the field used to determine when to display an animation.
`targetFieldChangeToValues`	string	Required. Values used to determine when to display an animation. When a user selects a value in `targetField` that matches a value stored in `targetFieldChangeToValues`, the animation is displayed.

Declarative Metadata Sample Definition

The following is an example of an AnimationRule component.

```
<?xml version="1.0" encoding="UTF-8"?>
<AnimationRule xmlns="http://soap.sforce.com/2006/04/metadata">
    <animationFrequency>Always</animationFrequency>
    <developerName>AnimationRule_DeveloperName</developerName>
    <isActive>true</isActive>
    <masterLabel>AnimationRule Master Label</masterLabel>
    <recordTypeContext>All</recordTypeContext>
    <recordTypeName>__MASTER__</recordTypeName>
    <sobjectType>Opportunity</sobjectType>
    <targetField>StageName</targetField>
   <targetFieldChangeToValues>Delivered, Negotiating, Closed Won</targetFieldChangeToValues>
</AnimationRule>
```

The following is an example `package.xml` that references the AnimationRule component.

```
<?xml version="1.0" encoding="UTF-8"?>
<Package xmlns="http://soap.sforce.com/2006/04/metadata">
    <types>
        <members>PathAssistant</members>
        <name>Settings</name>
    </types>
    <types>
        <members>AnimationRule_Developer_Name</members>
        <name>AnimationRule</name>
    </types>
    <version>46.0</version>
</Package>
```

Wildcard Support in the Manifest File

This metadata type supports the wildcard character `*` (asterisk) in the `package.xml` manifest file. For information about using the manifest file, see Deploying and Retrieving Metadata with the Zip File.

ArticleType

Represents the metadata associated with an article type.

All articles in Salesforce Knowledge are assigned to an *article type*. An article's type determines the type of content it contains, its appearance, and which users can access it. For example, a simple FAQ article type can have two custom fields, `Question` and `Answer`, where article managers enter data when creating or updating FAQ articles. A more complex article type can have dozens of fields organized into several sections. Using layouts and templates, administrators can structure the article type in the most effective way for its particular content. User access to article types is controlled by permissions. For each article type, an administrator can grant “Create,” “Read,” “Edit,” or “Delete” permissions to users. For example, the article manager can allow internal users to read, create, and edit FAQ article types, but let partner users only read FAQs. See “Knowledge Article Types” in the Salesforce online help and “Knowledge” in the *SOAP API Developer Guide*.

Declarative Metadata File Suffix and Directory Location

An ArticleType is defined as a custom object and is stored in the `objects` folder. ArticleTypes have a suffix `__kav` (instead of `__c` for custom objects). ArticleType field names have a suffix of `__c` like other custom objects, and must be dot-qualified with the name of the article type to which they belong. This is shown in the following sample `package.xml` file:

```
<?xml version="1.0" encoding="UTF-8"?>
<Package xmlns="http://soap.sforce.com/2006/04/metadata">
    <fullName>articlefilemetadata</fullName>
    <apiAccessLevel>Unrestricted</apiAccessLevel>
    <types>
        <members>newarticle__kav.description__c</members>
        <name>CustomField</name>
    </types>
    <types>
        <members>newarticle__kav</members>
        <name>CustomObject</name>
    </types>
</Package>
```

Version

ArticleTypes are available in API version 19.0 and later.

Fields

Field Name	Field Type	Description
`articleTypeChannelDisplay`	articleTypeChannelDisplay	Represents the article-type templates used to display an article in the various channels. See “Article Type Templates” in the Salesforce online help.
`deploymentStatus`	DeploymentStatus (enumeration of type string)	A string which represents the deployment status of a custom object or field. Valid values are: • `InDevelopment` • `Deployed`
`description`	string	A description of the article type. Maximum of 1000 characters.
`fields`	CustomField[]	Represents one or more fields in the article type.
`gender`	Gender	Indicates the gender of the noun that represents the object. This is used for languages where words need different treatment depending on their gender.
`label`	string	Label that represents the object throughout the Salesforce user interface.
`pluralLabel`	string	Plural version of the `label` value.

Field Name	Field Type	Description
`startsWith`	StartsWith (enumeration of type string)	Indicates whether the noun starts with a vowel, consonant, or is a special character. This is used for languages where words need different treatment depending on the first character. Valid values are listed in StartsWith.

articleTypeChannelDisplay

Determines the article-type templates that are used to display an article in its channels. Unless otherwise noted, all fields are createable, filterable, and nillable.

Field Name	Field Type	Description
`articleTypeTemplates`	articleTypeTemplates	Indicates which article-type template applies in the specified channel.

articleTypeTemplates

Sets the article-type template for a specific channel. If not specified, the default article-type template applies.

Field Name	Field Type	Description
`channel`	string	Specifies the channel where the article-type template applies: • `AllChannels`: all the available channels. • `App`: the Articles tab in Salesforce Knowledge. • `Pkb`: the public knowledge base. • `Csp`: the Customer Portal. • `Prm`: the partner portal.
`page`	string	Represents the name of the custom Visualforce page used as a custom article-type template. Use this field when you select `Page` in the template field.
`template`	string	Indicates the article-type template used for the specified channel: • `Page`: custom Visualforce page. When specifying this value, you must also set the `page` field with the Visualforce page name. • `Tab`: display the sections you defined in the layout as tabs. • `Toc`: display the sections you defined in the layout as table of content.

Declarative Metadata Sample Definitions

A sample article type definition follows:

```
<?xml version="1.0" encoding="UTF-8"?>
<CustomObject xmlns="http://soap.sforce.com/2006/04/metadata">
    <articleTypeChannelDisplay>
        <articleTypeTemplates>
```

```
            <channel>App</channel>
            <template>Tab</template>
        </articleTypeTemplates>
        <articleTypeTemplates>
            <channel>Prm</channel>
            <template>Tab</template>
        </articleTypeTemplates>
        <articleTypeTemplates>
            <channel>Csp</channel>
            <template>Tab</template>
        </articleTypeTemplates>
        <articleTypeTemplates>
            <channel>Pkb</channel>
            <template>Toc</template>
        </articleTypeTemplates>
    </articleTypeChannelDisplay>
    <deploymentStatus>Deployed</deploymentStatus>
    <description>Article type with custom fields</description>
    <fields>
        <fullName>description__c</fullName>
        <label>Description</label>
        <length>48</length>
        <type>Text</type>
    </fields>
    <label>newarticle</label>
    <pluralLabel>newarticles</pluralLabel>
</CustomObject>
```

Wildcard Support in the Manifest File

This metadata type supports the wildcard character `*` (asterisk) in the `package.xml` manifest file. For information about using the manifest file, see Deploying and Retrieving Metadata with the Zip File.

IN THIS SECTION:

ArticleType Layout

Represents the metadata associated with an article type page layout. Article type layouts determine which fields users can view and edit when entering data for an article, they also determine which sections appear when users view articles.

ChannelLayout

Represents the metadata associated with a communication channel layout. Communication channel layouts allow admins to share article contents inline into communication channels (for example, in email publishers, communities and sites, or social media publishers). Admins can create a list of fields for an article type or record type that they want to share for each communication channel. You can customize the order of the fields.

ArticleType CustomField

Represents the metadata associated with an article type custom field. Use this metadata type to create, update, or delete article type custom field definitions.

SEE ALSO:

ArticleType Layout

ArticleType CustomField

ArticleType Layout

Represents the metadata associated with an article type page layout. Article type layouts determine which fields users can view and edit when entering data for an article, they also determine which sections appear when users view articles.

The format of the article, for example whether layout sections display as subtabs or as a single page with links, is defined by the article-type template. Each article type has only one layout, but you can choose a different template for each of the article type's four channels. For more information, see "Knowledge Article Types" in the Salesforce online help and “Knowledge” in the *SOAP API Developer Guide*

File Suffix and Directory Location

ArticleType layouts are stored in the `layouts` directory of the corresponding package directory. The prefix must match with the article type API name. The extension is `.layout`.

Version

ArticleType layouts are available in API version 19.0 and later.

Fields

Field Name	Field Type	Description
`layoutSections`	LayoutSection[]	The main sections of the layout containing the article fields. The order here determines the layout order.

LayoutSection

LayoutSection represents a section of an ArticleType layout.

Field Name	Field Type	Description
`customLabel`	boolean	Indicates if this section's label is custom or standard (built-in). Custom labels can be any text, but must be translated. Standard labels have a predefined set of valid values, for example 'System Information', which are automatically translated.
`label`	string	The label; either standard or custom, based on the `customLabel` flag.

Field Name	Field Type	Description
`layoutColumns`	LayoutColumn[]	The columns of the layout, depending on the style. Salesforce Knowledge only supports one column in article type layouts.
`style`	LayoutSectionStyle (enumeration of type string)	The style of the layout. Salesforce Knowledge only supports the value `OneColumn` which displays a one column page.

LayoutColumn

LayoutColumn represents the items in a column within a layout section.

Field Name	Field Type	Description
`layoutItems`	LayoutItem[]	The individual items within a column (ordered from top to bottom).

LayoutItem

LayoutItem represents the valid values that define a layout item.

Field Name	Field Type	Description
`field`	string	The field name reference, for example `MyField__c`.

Declarative Metadata Sample Definition

The following is the definition of an ArticleType page layout:

```
<?xml version="1.0" encoding="UTF-8"?>
<Layout xmlns="http://soap.sforce.com/2006/04/metadata">
    <layoutSections>
        <customLabel>true</customLabel>
        <label>Description</label>
        <layoutColumns>
            <layoutItems>
                <field>description__c</field>
            </layoutItems>
            <layoutItems>
                <field>dateTime__c</field>
            </layoutItems>
        </layoutColumns>
        <style>OneColumn</style>
    </layoutSections>
    <layoutSections>
        <label>Data Sheet</label>
        <layoutColumns>
            <layoutItems>
                <field>file__c</field>
            </layoutItems>
        </layoutColumns>
        <style>OneColumn</style>
    </layoutSections>
</Layout>
```

SEE ALSO:

ArticleType

ArticleType CustomField

ChannelLayout

Represents the metadata associated with a communication channel layout. Communication channel layouts allow admins to share article contents inline into communication channels (for example, in email publishers, communities and sites, or social media publishers). Admins can create a list of fields for an article type or record type that they want to share for each communication channel. You can customize the order of the fields.

File Suffix and Directory Location

Channel layout components have the suffix `.channelLayout` and are stored in the `channelLayouts` folder of the corresponding package directory. The prefix must match with the article type API name. In Lightning Knowledge, the prefix must match the API name for the knowledge object.

Version

Channel layout components are available in API version 32.0 and later.

Fields

Field Name	Field Type	Description
`doesExcludeFieldLabels`	boolean	Indicates whether field labels are excluded from the field contents in the communication channels where this layout applies (`true`) or not (`false`). The default is `false`, meaning field labels are inserted. Available when Lightning Knowledge is enabled in API version 48.0 and later.
`doesExcludeFiles`	boolean	Indicates whether related files are left off emails (`true`) or attached to emails (`false`). The default is `false`, meaning related files are attached. Available when Lightning Knowledge is enabled in API version 48.0 and later.
`enabledChannels`	string	The communication channels where this layout applies. In API version 32.0 to 46.0, the only valid value is `Email`. When Lightning Knowledge is enabled in API version 47.0 and later, `Chat`, `Messaging`, and `Social` are additional valid values.
`label`	string	Required. The label for this configuration.
`layoutItems`	ChannelLayoutItem on page 158	The article fields contained in the layout. The order here determines the field order.

Field Name	Field Type	Description
`recordType`	string	The name of the record type that the channel layout applies to. The default is the master record type. Available in API version 41.0 and later.

ChannelLayoutItem

Field Name	Field Type	Description
`field`	string	Required. Name of the field. The format is *`ArticleTypeName.FieldName`* or, in Lightning Knowledge, *`KnowledgeBaseName.FieldName`*.

Declarative Metadata Sample Definition

The following is an example of a ChannelLayout component.

```
<?xml version="1.0" encoding="UTF-8"?>
<ChannelLayout xmlns="http://soap.sforce.com/2006/04/metadata">
    <label>Layout for Email</label>
    <layoutItems>
        <field>Knowledge.Question</field>
    </layoutItems>
    <layoutItems>
        <field>Knowledge.Answer</field>
    </layoutItems>
    <enabledChannels>Email</enabledChannels>
    <enabledChannels>Social</enabledChannels>
    <enabledChannels>Chat</enabledChannels>
    <doesExcludeFiles>false</doesExcludeFiles>
    <doesExcludeFieldLabels>true</doesExcludeFieldLabels>
</ChannelLayout>
```

The following is an example `package.xml` that references the previous definition.

```
<?xml version="1.0" encoding="UTF-8"?>
<Package xmlns="http://soap.sforce.com/2006/04/metadata">
    <types>
        <members>*</members>
        <name>ChannelLayout</name>
    </types>
    <version>41.0</version>
</Package>
```

ArticleType CustomField

Represents the metadata associated with an article type custom field. Use this metadata type to create, update, or delete article type custom field definitions.

This type extends the Metadata metadata type and inherits its `fullName` field.

Always specify the full name whenever you create or update a custom field. For example, a custom field on a custom object:

```
MyArticleType__kav.MyCustomField__c
```

Declarative Metadata File Suffix and Directory Location

Custom fields are defined as part of the article type. ArticleType field names have a suffix of `__c` like other custom objects, and must be dot-qualified with the name of the article type to which they belong. See ArticleType for more information.

Retrieving Custom Fields on Custom or Standard Objects

When you retrieve a custom or standard object, you return everything associated with the object. However, you can also retrieve only the custom fields for an object by explicitly naming the object and fields in `package.xml`. The following definition in `package.xml` retrieves the files `objects/MyCustomObject__c.object`, `objects/Account.object__c.object`, and `objects/MyArticleType__kav.object`, each containing one custom field definition.

```
<types>
    <members>MyCustomObject__c.MyCustomField__c</members>
    <members>Account.MyCustomAccountField__c</members>
    <members>MyArticleType__kav.MyOtherCustomField__c</members>
    <name>CustomField</name>
</types>
```

Version

ArticleTypes custom fields are available in API version 19.0 and later.

Fields for ArticleType

Unless otherwise noted, all fields are createable, filterable, and nillable.

Note: If you create a knowledge validation rule, the errors always display at the top of the page, even if you add it beside the field. Therefore, write the errors descriptively so authors know how to satisfy the validation rule. For example, identify which field is causing the error. The Salesforce Classic user interface does not support field level error messages for articles.

Field Name	Field Type	Description
`defaultValue`	string	If specified, represents the default value of the field.
`deleteConstraint`	DeleteConstraint (enumeration of type string)	Provides deletion options for lookup relationships. Valid values are: **`SetNull`** This is the default. If the lookup record is deleted, the lookup field is cleared. **`Restrict`** Prevents the record from being deleted if it's in a lookup relationship.

Field Name	Field Type	Description
		Cascade Deletes the lookup record as well as associated lookup fields. For more information on lookup relationships, see "Object Relationships" in the Salesforce Help.
description	string	Description of the field.
formula	string	If specified, represents a formula on the field.
formulaTreatBlankAs	TreatBlanksAs (enumeration of type string)	Indicates how to treat blanks in a formula. Valid values are BlankAsBlank and BlankAsZero.
fullName	string	Inherited from Metadata, this field is not defined in the WSDL for this metadata type. It must be specified when creating, updating, or deleting. See createMetadata() to see an example of this field specified for a call. This value cannot be null.
inlineHelpText	string	Represents the content of field-level help. For more information, see "Define Field-Level Help" in the Salesforce Help.
label	string	Label for the field. You cannot update the label for standard fields in Article Type such as Title, UrlName, Summary, etc.
length	int	Length of the field.
picklist	Picklist (Including Dependent Picklist)	(**Deprecated.** Use this field in API version 37.0 and earlier only. In later versions, use valueSet instead.) If specified, the field is a picklist, and this field enumerates the picklist values and labels.
referenceTo	string	If specified, indicates a reference this field has to another object.
relationshipLabel	string	Label for the relationship.
relationshipName	string	If specified, indicates the value for one-to-many relationships. For example, in the object MyObject that had a relationship to YourObject, the relationship name might be YourObjects.
required	boolean	Indicates whether the field requires a value on creation (true) or not (false).
type	FieldType	Required. Indicates the field type for the field. Valid values are: • Checkbox available in version 30.0 and later • Currency • ArticleCurrency • Date • DateTime

Field Name	Field Type	Description
		- `Email` - `File` - `Formula` - `Html` - `Lookup` - `Number` - `Percent` - `Phone` - `Picklist` - `DependentPicklist` - `MultiselectPicklist` - `Text` - `TextArea` - `LongTextArea` - `URL`
`visibleLines`	int	Indicates the number of lines displayed for the field.

Declarative Metadata Sample Definition

```
<?xml version="1.0" encoding="UTF-8"?>
<CustomObject xmlns="http://soap.sforce.com/2006/04/metadata">
    <fields>
        <fullName>Comments__c</fullName>
        <description>add your comments about this object here</description>
        <label>Comments</label>
        <length>32000</length>
        <type>LongTextArea</type>
        <visibleLines>30</visibleLines>
    </fields>
</CustomObject>
```

SEE ALSO:

ArticleType

ArticleType Layout

ApexClass

Represents an Apex class. An Apex class is a template or blueprint from which Apex objects are created. Classes consist of other classes, user-defined methods, variables, exception types, and static initialization code.

For more information, see the *Lightning Platform Apex Code Developer's Guide*. This type extends the MetadataWithContent metadata type and inherits its `content` and `fullName` fields.

Note: By default, you can't deploy updates to an Apex class if there are one or more active jobs for that class. To deploy updates in this case, do one of the following.

- Cancel Apex jobs before deploying changes to Apex code. Reschedule the jobs after the deployment.
- Enable deployments with Apex jobs in the Salesforce user interface in the Deployment Settings page.

Supported Calls

All Metadata API calls except CRUD-Based Calls, which prevents deployment outside of proper deployment lifecycle and test-execution constraints.

Declarative Metadata File Suffix and Directory Location

The file suffix is `.cls` for the class file. The accompanying metadata file is named `*ClassName*-meta.xml`.

Apex classes are stored in the `classes` folder in the corresponding package directory.

Version

Apex classes are available in API version 10.0 and later.

Fields

This metadata type contains the following fields:

Field Name	Field Type	Description
`apiVersion`	double	The API version for this class. Every class has an API version specified at creation.
`content`	base64	The Apex class definition. Base 64-encoded binary data. Prior to making an API call, client applications must encode the binary attachment data as base64. Upon receiving a response, client applications must decode the base64 data to binary. This conversion is usually handled for you by a SOAP client. This field is inherited from the MetadataWithContent component.
`fullName`	string	The Apex class name. The name can only contain characters, letters, and the underscore (_) character, must start with a letter, and cannot end with an underscore or contain two consecutive underscore characters. This field is inherited from the Metadata component.
`packageVersions`	PackageVersion[]	The list of installed managed package versions that are referenced by this Apex class. For more information about managed packages, see the Lightning Platform Quick Reference for Developing Packages. For more information about package versions, see "About Package Versions" in the Salesforce online help. This field is available in API version 16.0 and later.

Field Name	Field Type	Description
`status`	ApexCodeUnitStatus (enumeration of type string)	The current status of the Apex class. The following string values are valid: • `Active` - The class is active. • `Deleted` - The class is marked for deletion. This is useful for managed packages, because it allows a class to be deleted when a managed package is updated. Note: ApexCodeUnitStatus includes an `Inactive` option, but it is only supported for ApexTrigger; it is not supported for ApexClass.

PackageVersion

PackageVersion identifies a version of a managed package. A package version is a number that identifies the set of components uploaded in a package. The version number has the format *`majorNumber.minorNumber.patchNumber`* (for example, 2.1.3). The major and minor numbers increase to a chosen value during every major release. The *`patchNumber`* is generated and updated only for a patch release. It is available in API version 16.0 and later.

Field Name	Field Type	Description
`namespace`	string	Required. In a packaging context, a namespace prefix is a one to 15-character alphanumeric identifier that distinguishes your package and its contents from packages of other developers on AppExchange. Namespace prefixes are case-insensitive. For example, ABC and abc are not recognized as unique. Your namespace prefix must be globally unique across all Salesforce organizations. It keeps your managed package under your control exclusively. Salesforce automatically prepends your namespace prefix, followed by two underscores ("__"), to all unique component names in your Salesforce organization. A unique package component is one that requires a name that no other component has within Salesforce, such as custom objects, custom fields, custom links, s-controls, and validation rules. For more information about namespaces, see "Register a Namespace Prefix" in the Salesforce online help.
`majorNumber`	int	Required. The major number of the package version. A package version number has a *`majorNumber.minorNumber`* format.
`minorNumber`	int	Required. The minor number of the package version. A package version number has a *`majorNumber.minorNumber`* format.

Declarative Metadata Sample Definition

The following sample creates the `MyhelloWorld.cls` class, and the corresponding `MyHelloWorld.cls-meta.xml` metadata file.

`MyHelloWorld.cls` file:

```
public class MyHelloWorld {
// This method updates the Hello field on a list
// of accounts.
public static void addHelloWorld(Account[] accs){
 for (Account a:accs){
  if (a.Hello__c != 'World')
  a.Hello__c = 'World';
  }
 }
}
```

MyHelloWorld.cls-meta.xml:

```
<?xml version="1.0" encoding="UTF-8"?>
<ApexClass xmlns="http://soap.sforce.com/2006/04/metadata">
    <apiVersion>48.0</apiVersion>
</ApexClass>
```

Wildcard Support in the Manifest File

This metadata type supports the wildcard character * (asterisk) in the package.xml manifest file. For information about using the manifest file, see Deploying and Retrieving Metadata with the Zip File.

SEE ALSO:

ApexTrigger

ApexComponent

Represents a Visualforce component.

For more information, see “Visualforce” in the Salesforce online help. This type extends the MetadataWithContent metadata type and inherits its content and fullName fields.

Declarative Metadata File Suffix and Directory Location

The file suffix is .component for the page file. The accompanying metadata file is named *ComponentName*-meta.xml.

Visualforce components are stored in the components folder in the corresponding package directory.

Version

Visualforce components are available in API version 12.0 and later.

Fields

This metadata type contains the following fields:

Field Name	Field Type	Description
apiVersion	double	The API version for this Visualforce component. Every component has an API version specified at creation. This field is available in API version 16.0 and later.

Field Name	Field Type	Description
`content`	base64Binary	The component content. Base 64-encoded binary data. Prior to making an API call, client applications must encode the binary attachment data as base64. Upon receiving a response, client applications must decode the base64 data to binary. This conversion is usually handled for you by a SOAP client. This field is inherited from the MetadataWithContent component.
`description`	string	A description of what the component does.
`fullName`	string	The component developer name used as a unique identifier for API access. The `fullName` can contain only underscores and alphanumeric characters. It must be unique, begin with a letter, not include spaces, not end with an underscore, and not contain two consecutive underscores. This field is inherited from the Metadata component.
`label`	string	Required. The label for this component.
`packageVersions`	PackageVersion[]	The list of installed managed package versions that are referenced by this Visualforce component. Note: Package components and Visualforce custom component are distinct concepts. A package is comprised of many elements, such as custom objects, Apex classes and triggers, and custom pages and components. For more information about managed packages, see the Lightning Platform Quick Reference for Developing Packages. For more information about package versions, see "About Package Versions" in the Salesforce online help. This field is available in API version 16.0 and later.

Wildcard Support in the Manifest File

This metadata type supports the wildcard character `*` (asterisk) in the `package.xml` manifest file. For information about using the manifest file, see Deploying and Retrieving Metadata with the Zip File.

SEE ALSO:

ApexPage

ApexPage

Represents a Visualforce page.

For more information, see "Visualforce" in the Salesforce online help. This type extends the MetadataWithContent metadata type and inherits its `content` and `fullName` fields.

Declarative Metadata File Suffix and Directory Location

The file suffix is `.page` for the page file. The accompanying metadata file is named `*PageName*-meta.xml`.

Visualforce pages are stored in the `pages` folder in the corresponding package directory.

Version

Visualforce pages are available in API version 11.0 and later.

Fields

This metadata type contains the following fields:

Field Name	Field Type	Description
`apiVersion`	double	Required. The API version for this page. Every page has an API version specified at creation. This field is available in API version 15.0 and later. If you set this field to a number lower than 15.0, it will be changed to 15.0.
`content`	base64Binary	The page content. Base 64-encoded binary data. Prior to making an API call, client applications must encode the binary attachment data as base64. Upon receiving a response, client applications must decode the base64 data to binary. This conversion is usually handled for you by a SOAP client. This field is inherited from the MetadataWithContent component.
`description`	string	A description of what the page does.
`fullName`	string	The page developer name used as a unique identifier for API access. The `fullName` can contain only underscores and alphanumeric characters. It must be unique, begin with a letter, not include spaces, not end with an underscore, and not contain two consecutive underscores. This field is inherited from the Metadata component.
`availableInTouch`	boolean	Indicates if Visualforce tabs associated with the Visualforce page can be used in the Salesforce app. (Use of this field for Salesforce Touch is deprecated.). This field is available in API version 27.0 and later. Standard object tabs that are overridden with a Visualforce page aren't supported in the Salesforce app, even if you set this field for the page. The default page for the object is displayed instead of the Visualforce page.
`confirmationTokenRequired`	boolean	Indicates whether `GET` requests for the page require a CSRF confirmation token. This field is available in API version 28.0 and later. If you change this field's value from `false` to `true`, links to the page require a CSRF token to be added to them, or the page will be inaccessible.
`label`	string	Required. The label for this page.

Field Name	Field Type	Description
packageVersions	PackageVersion[]	The list of installed managed package versions that are referenced by this Visualforce page. For more information about managed packages, see the Lightning Platform Quick Reference for Developing Packages. For more information about package versions, see “About Package Versions” in the Salesforce online help. This field is available in API version 16.0 and later.

Declarative Metadata Sample Definition

The following sample creates the MyPage.page page, and the corresponding MyPage.page-meta.xml metadata file.

SampleApexPage.page file:

```
<apex:page>
<h1>Congratulations</h1>
This is your new Page.
</apex:page>
```

SampleApexPage.page-meta.xml:

```
<?xml version="1.0" encoding="UTF-8"?>
<ApexPage xmlns="http://soap.sforce.com/2006/04/metadata">
    <description>This is a sample Visualforce page.</description>
    <label>SampleApexPage</label>
</ApexPage>
```

Wildcard Support in the Manifest File

This metadata type supports the wildcard character * (asterisk) in the package.xml manifest file. For information about using the manifest file, see Deploying and Retrieving Metadata with the Zip File.

SEE ALSO:

ApexComponent

ApexTestSuite

Represents a suite of Apex test classes to include in a test run.

File Suffix and Directory Location

ApexTestSuite components have the suffix .testSuite and are stored in the testSuites folder.

Version

ApexTestSuite components are available in API version 38.0 and later.

Fields

Field Name	Field Type	Description
`testClassName`	string[]	A list of Apex test classes, specified by name, to include in this test suite.

Declarative Metadata Sample Definition

To include namespaced tests in an Apex test suite, specify each namespace individually. Local Apex tests consist of all tests in the org that don’t originate from managed packages.

```
<?xml version="1.0" encoding="UTF-8"?>
<ApexTestSuite xmlns="http://soap.sforce.com/2006/04/metadata">
   <testClassName>LocalTestClass</testClassName>
      <!-- LocalTestClass adds the test class named LocalTestClass. -->
   <testClassName>A*Class</testClassName>
      <!-- A*Class adds AClass, AnotherClass, AwesomeClass, and so on. -->
   <testClassName>Namespace1.NamespacedTestClass</testClassName>
   <testClassName>*</testClassName> <!-- Adds all local tests. -->
   <testClassName>Namespace1.*</testClassName> <!-- Adds all tests in Namespace1. -->
   <testClassName>Namespace2.*</testClassName> <!-- Adds all tests in Namespace2. -->
</ApexTestSuite>
```

These syntaxes are supported in `package.xml`. If the test classes in your suites are already present in the target org, you can omit the `ApexClass` type in `package.xml`.

```
<?xml version="1.0" encoding="UTF-8"?>
<Package xmlns="http://soap.sforce.com/2006/04/metadata">
   <types>
      <members>*</members>
      <name>ApexClass</name>
   </types>
   <types>
      <members>*</members>
      <name>ApexTestSuite</name>
   </types>
   <version>38.0</version>
</Package>
```

```
<?xml version="1.0" encoding="UTF-8"?>
<Package xmlns="http://soap.sforce.com/2006/04/metadata">
   <types>
      <members>*</members>
      <name>ApexClass</name>
   </types>
   <types>
      <members>Suite1</members>
      <members>Suite2</members>
      <name>ApexTestSuite</name>
   </types>
```

```
    <version>38.0</version>
</Package>
```

Wildcard Support in the Manifest File

This metadata type supports the wildcard character `*` (asterisk) in the `package.xml` manifest file. For information about using the manifest file, see Deploying and Retrieving Metadata with the Zip File.

ApexTrigger

Represents an Apex trigger. A trigger is Apex code that executes before or after specific data manipulation language (DML) events occur, such as before object records are inserted into the database, or after records have been deleted.

For more information, see “Manage Apex Triggers” in the Salesforce online help. This type extends the MetadataWithContent metadata type and inherits its `content` and `fullName` fields.

Supported Calls

All Metadata API calls except CRUD-Based Calls, which prevents deployment outside of proper deployment lifecycle and test-execution constraints.

Declarative Metadata File Suffix and Directory Location

The file suffix is `.trigger` for the trigger file. The accompanying metadata file is named `*TriggerName*-meta.xml`.

Apex triggers are stored in the `triggers` folder in the corresponding package directory.

Version

Triggers are available in API version 10.0 and later.

Fields

This metadata type contains the following fields:

Field Name	Field Type	Description
`apiVersion`	double	Required. The API version for this trigger. Every trigger has an API version specified at creation.
`content`	base64	The Apex trigger definition. This field is inherited from the MetadataWithContent component.
`fullName`	string	The Apex trigger name. The name can only contain characters, letters, and the underscore (_) character, must start with a letter, and cannot end with an underscore or contain two consecutive underscore characters. This field is inherited from the Metadata component.

Field Name	Field Type	Description
packageVersions	PackageVersion[]	The list of installed managed package versions that are referenced by this Apex trigger. For more information about managed packages, see the Lightning Platform Quick Reference for Developing Packages. For more information about package versions, see "About Package Versions" in the Salesforce online help. This field is available in API version 16.0 and later.
status	ApexCodeUnitStatus (enumeration of type string)	Required. The current status of the Apex trigger. The following string values are valid: • Active - The trigger is active. • Inactive - The trigger is inactive, but not deleted. • Deleted - The trigger is marked for deletion. This is useful for managed packages, because it allows a trigger to be deleted when a managed package is updated.

Declarative Metadata Sample Definition

The following sample creates the MyhelloWorld.trigger trigger, and the corresponding MyHelloWorld.trigger-meta.xml metadata file.

MyHelloWorld.trigger file:

```
trigger helloWorldAccountTrigger on Account (before insert) {

  Account[] accs = Trigger.new;

   MyHelloWorld.addHelloWorld(accs);
}
```

MyHelloWorld.trigger-meta.xml:

```
<?xml version="1.0" encoding="UTF-8"?>
<ApexTrigger xmlns="http://soap.sforce.com/2006/04/metadata">
    <apiVersion>48.0</apiVersion>
</ApexTrigger>
```

Wildcard Support in the Manifest File

This metadata type supports the wildcard character * (asterisk) in the package.xml manifest file. For information about using the manifest file, see Deploying and Retrieving Metadata with the Zip File.

SEE ALSO:

ApexClass

AppMenu

Represents the app menu or the Salesforce mobile navigation menu.

File Suffix and Directory Location

Each AppMenu component gets stored in a single file in the folder of the corresponding package directory. The filename uses the format *`Feature`*`.appMenu`.

- There's one app switcher app menu file stored in a file named `AppSwitcher.appMenu`.
- There's one Salesforce app menu file stored in a file named `Salesforce1.appMenu`.

These two files are located in the `appMenus` folder. The `.appMenu` files are different from other named components, as there's only one file for each AppMenu component. App menu files can't be created or deleted.

Version

AppMenu components are available in API version 30.0 and later.

Fields

Field Name	Field Type	Description
`appMenuItems`	AppMenuItem[]	A list of menu items in the app menu.

AppMenuItem

Represents a menu item in the app menu.

Field Name	Field Type	Description
`name`	string	The API name of the item.
`type`	string	The type of application represented by this item. Acceptable values for `AppSwitcher.appMenu` are: • *`ConnectedApp`* • *`CustomApplication`* • *`ServiceProvider`* Acceptable values for `Salesforce1.appMenu` are: • *`CustomApplication`* • *`CustomTab`* • *`StandardAppMenuItem`*. The name for this item can be: – *`MyDay`* – *`Feed`*

Field Name	Field Type	Description
		- *Tasks* - *Dashboards* - *Search* - *People* (available only when Chatter is enabled) - *Groups* (available only when Chatter is enabled)

Declarative Metadata Sample Definition

The following is an example of an AppSwitcher.appMenu file.

```
<?xml version="1.0" encoding="UTF-8"?>
<AppMenu xmlns="http://soap.sforce.com/2006/04/metadata">
   <appMenuItems>
      <appMenuItem>
         <name>standard__Sales</name>
         <type>CustomApplication</type>
      </appMenuItem>
      <appMenuItem>
         <name>standard__Support</name>
         <type>CustomApplication</type>
      </appMenuItem>
      <appMenuItem>
         <name>CustomApp1</name>
         <type>CustomApplication</type>
      </appMenuItem>
      <appMenuItem>
         <name>CustomApp2</name>
         <type>CustomApplication</type>
      </appMenuItem>
      <appMenuItem>
         <name>ConnectedApp1</name>
         <type>ConnectedApp</type>
      </appMenuItem>
   </appMenuItems>
</AppMenu>
```

The following is an example `package.xml` that references the previous definition.

```
<?xml version="1.0" encoding="UTF-8"?>
<Package xmlns="http://soap.sforce.com/2006/04/metadata">
   <types>
      <members>AppSwitcher</members>
      <name>AppMenu</name>
   </types>
```

The following is an example of a `Salesforce1.appMenu` component.

```
<?xml version="1.0" encoding="UTF-8"?>
<AppMenu xmlns="http://soap.sforce.com/2006/04/metadata">
   <appMenuItems>
```

```
        <appMenuItem>
            <name>StandardItem1</name>
            <type>StandardAppMenuItem</type>
        </appMenuItem>
        <appMenuItem>
            <name>StandardItem2</name>
            <type>StandardAppMenuItem</type>
        </appMenuItem>
        <appMenuItem>
            <name>StandardItem3</name>
            <type>StandardAppMenuItem</type>
        </appMenuItem>
        <appMenuItem>
            <name>CustomTab1</name>
            <type>CustomTab</type>
        </appMenuItem>
    </appMenuItems>
</AppMenu>
```

The following is an example `package.xml` that references the previous definition.

```
<?xml version="1.0" encoding="UTF-8"?>
<Package xmlns="http://soap.sforce.com/2006/04/metadata">
    <types>
        <members>Salesforce1</members>
        <name>AppMenu</name>
    </types>
```

The following is an example of a package manifest used to deploy or retrieve all the available app menu metadata for an organization, using a wildcard:

```
<?xml version="1.0" encoding="UTF-8"?>
<Package xmlns="http://soap.sforce.com/2006/04/metadata">
    <types>
        <members>*</members>
        <name>AppMenu</name>
    </types>
    <version>30.0</version>
</Package>
```

Usage

Use `AppSwitcher.appMenu` to reorder the list of menu items that appears in the app menu. You can't add app menu items to or remove app menu items from `AppSwitcher.appMenu`.

Use `Salesforce1.appMenu` to customize the list of menu items that appears in the Salesforce navigation menu by reordering, adding, or removing the app menu items.

Wildcard Support in the Manifest File

This metadata type supports the wildcard character `*` (asterisk) in the `package.xml` manifest file. For information about using the manifest file, see Deploying and Retrieving Metadata with the Zip File.

AppointmentSchedulingPolicy

Represents a set of rules for scheduling appointments using Lightning Scheduler. This type extends the Metadata metadata type and inherits its `fullName` field.

File Suffix and Directory Location

AppointmentSchedulingPolicy components have the suffix `.policy` and are stored in the `appointmentSchedulingPolicies` folder.

Version

AppointmentSchedulingPolicy components are available in API version 47.0 and later.

Special Access Rules

You must have the ViewSetup and CustomizeApplication user permissions to access the AppointmentSchedulingPolicy type.

Fields

Field Name	Field Type	Description
`appointmentStartTimeInterval`	picklist	Required. The proposed time interval in minutes between appointment start times. For example, if you set the interval to `15`, appointments can then begin at the top of the hour and at 15-minute intervals thereafter (10:00 AM, 10:15 AM, 10:30 AM, and so on). Valid values are: - `5` - `10` - `15` - `20` - `30` - `60`
`masterLabel`	string	Required. The label for the appointment scheduling policy.
`shouldConsiderCalendarEvents`	boolean	Required. Indicates whether to consider events on the Salesforce calendar to determine the availability of service resources to be assigned to appointments (`true`) or not (`false`).
`shouldEnforceExcludedResource`	boolean	Required. Indicates whether this appointment scheduling policy prevents excluded service resources from being assigned to appointments (`true`) or not (`false`).
`shouldEnforceRequiredResource`	boolean	Required. Indicates whether this appointment scheduling policy allows only required service resources to be assigned to appointments (`true`) or not (`false`).

Field Name	Field Type	Description
`shouldMatchSkill`	boolean	Required. Indicates whether this appointment scheduling policy allows only required service resources who have certain skills to be assigned to appointments (`true`) or not (`false`).
`shouldMatchSkillLevel`	boolean	Required. Indicates whether this appointment scheduling policy allows only required service resources who have certain skills and skill levels to be assigned to appointments (`true`) or not (`false`).
`shouldRespectVisitingHours`	boolean	Required. Indicates whether this appointment scheduling policy prevents users from scheduling appointments outside of an account's visiting hours (`true`) or not (`false`).
`shouldUsePrimaryMembers`	boolean	Required. Indicates whether this appointment scheduling policy allows only service resources who are primary members of a service territory to be assigned to appointments (`true`) or not (`false`).
`shouldUseSecondaryMembers`	boolean	Required. Indicates whether this appointment scheduling policy allows service resources who are secondary members of a service territory to be assigned to appointments (`true`) or not (`false`).

Declarative Metadata Sample Definition

The following is an example of an appointmentSchedulingPolicy component.

```
<?xml version="1.0" encoding="UTF-8"?>
<AppointmentSchedulingPolicy xmlns="http://soap.sforce.com/2006/04/metadata">
  <appointmentStartTimeInterval>15</appointmentStartTimeInterval>
  <masterLabel>Default Appointment Scheduling Policy</masterLabel>
  <shouldConsiderCalendarEvents>true</shouldConsiderCalendarEvents>
  <shouldEnforceExcludedResource>true</shouldEnforceExcludedResource>
  <shouldEnforceRequiredResource>true</shouldEnforceRequiredResource>
  <shouldMatchSkill>true</shouldMatchSkill>
  <shouldMatchSkillLevel>false</shouldMatchSkillLevel>
  <shouldRespectVisitingHours>true</shouldRespectVisitingHours>
  <shouldUsePrimaryMembers>true</shouldUsePrimaryMembers>
  <shouldUseSecondaryMembers>true</shouldUseSecondaryMembers>
</AppointmentSchedulingPolicy>
```

The following is an example `package.xml` that references the previous definition.

```
<?xml version="1.0" encoding="UTF-8"?>
 <Package xmlns="http://soap.sforce.com/2006/04/metadata">
  <types>
   <members>sample</members>
   <name>AppointmentSchedulingPolicy</name>
  </types>
  <version>47.0</version>
 </Package>
```

Wildcard Support in the Manifest File

This metadata type supports the wildcard character `*` (asterisk) in the `package.xml` manifest file. For information about using the manifest file, see Deploying and Retrieving Metadata with the Zip File.

ApprovalProcess

Represents the metadata associated with an approval process. An approval process automates how records are approved in Salesforce. An approval process specifies each step of approval, including who to request approval from and what to do at each point of the process.

This type extends the Metadata metadata type and inherits its `fullName` field.

Note:

- To use approval processes on Salesforce Knowledge articles with the Metadata API, the article type must be deployed. For article version (_kav) in approval processes, the supported action types are: Knowledge Action, Email Alert, Field Update, and Outbound Message.
- Send actions and approval processes for email drafts aren't supported in the Metadata API.
- The metadata doesn't include the order of active approval processes. You might need to reorder the approval processes in the destination org after deployment.
- Before you implement an approval process for your organization, see "Considerations for Approvals" in the Salesforce Help.

File Suffix and Directory Location

ApprovalProcess components have the suffix `.approvalProcess` and are stored in the `approvalProcesses` folder.

Version

ApprovalProcess components are available in API version 28.0 and later.

Fields

Field Name	Field Type	Description
`active`	boolean	Required. Whether the approval process is active. After an approval process is activated, you can't add, delete, or change the order of the steps or change its reject or skip behavior, even if the process is inactive.
`allowRecall`	boolean	Whether to allow submitters to recall approval requests. If set to `false`, only administrators can recall approval requests.
`allowedSubmitters`	ApprovalSubmitter[]	Required. An array of users who are allowed to submit records for approval.

Field Name	Field Type	Description
approvalPageFields	ApprovalPageField	Specifies which fields to display on the approval page, where the approver goes to approve or reject the record. By default, the approval page displays the following: • Name field • Owner field (except for child objects) If you enable notifications in the Salesforce mobile app, keep in mind that approvers may view this list of fields on a mobile device. Select only the fields necessary for users to decide whether to approve or reject records.
approvalStep	ApprovalStep[]	An array of approval step definitions.
description	string	Describes the approval process.
emailTemplate	string	Specifies which Classic email template to use for approval requests. If not specified, the default email template is used. Lightning email templates aren't packageable. We recommend using a Classic email template. When an approval process assigns an approval request to a user, Salesforce sends the user an approval request email.
enableMobileDeviceAccess	boolean	Whether users can access an external version of the approval page from any browser, including those on mobile devices, without logging in to Salesforce. Corresponds to Security Settings in the user interface. If set to true, approval steps can't have approvers of type adhoc. If set to false, approvers must log in to Salesforce to access the approval page.
entryCriteria	ApprovalEntryCriteria	Determines which records can enter the approval process. Exclude this field to allow all records to enter the approval process.
finalApprovalActions	ApprovalAction	Specifies which workflow actions to execute when all required approvals have been given for a record.
finalApprovalRecordLock	boolean	Whether to keep the record locked after it receives all necessary approvals. Default: false.
finalRejectionActions	ApprovalAction	Specifies which workflow actions to execute after a record enters the final rejection state.
finalRejectionRecordLock	boolean	Whether to keep the record locked after it's finally rejected. Default: false.
initialSubmissionActions	ApprovalAction	Specifies which workflow actions to execute when a record is initially submitted for approval.

Field Name	Field Type	Description
`label`	string	Required. Name of the approval process.
`nextAutomatedApprover`	NextAutomatedApprover	Specifies a standard or custom user hierarchy field that can be used to automatically assign the approver for an approval step. If you exclude this field, then no approval step can use a user hierarchy field to automatically assign the approver.
`postTemplate`	string	Post template to use for Approvals in Chatter. Chatter post approval notifications are only available for approval processes associated with an object that has been enabled for feed tracking.
`recallActions`	ApprovalAction	Specifies which workflow actions to execute when a pending approval request is withdrawn.
`recordEditability`	RecordEditabilityType (enumeration of type string)	Specifies which users can edit records that are pending approval. When a record is submitted for approval, it is automatically locked to prevent other users from editing it during the approval process. Valid values are: • `AdminOnly`—Records pending approval can be edited by: – Users with the “Modify All Data” permission – Users with the “Modify All” object-level permission for the given object • `AdminOrCurrentApprover`—Records pending approval can be edited by: – Users with the “Modify All Data” permission – Users with the “Modify All” object-level permission for the given object – The assigned approver, who must have edit access to the record through user permissions and the organization-wide sharing defaults for the given object
`showApprovalHistory`	boolean	Whether to add the Approval History related list to the approval page, which is where the approver can view the approval request details and approve or reject the record. The Approval History related list tracks a record through the approval process. If you also want to add the Approval History related list to record detail and edit pages, use the Salesforce user interface to customize the page layouts for the given object.

ApprovalSubmitter

Represents a user or set of users who can submit records for approval.

<table>
<tr><th>Field Name</th><th>Field Type</th><th>Description</th></tr>
<tr><td><code>submitter</code></td><td>string</td><td>Identifies a specific user or set of users who can submit records for approval. This field is required, except when the following types are specified and the <code>submitter</code> field is ignored:
• <code>owner</code>
• <code>creator</code>
• <code>allInternalUsers</code>
Example:<pre>
<allowedSubmitters>
    <type>allInternalUsers</type>
</allowedSubmitters>
<allowedSubmitters>
    <submitter>myGroup</submitter>
    <type>group</type>
</allowedSubmitters>
</pre></td></tr>
<tr><td><code>type</code></td><td>ProcessSubmitterType (enumeration of type string)</td><td>Required. Type of user or set of users who can submit records for approval. Valid values are:
• <code>group</code>
• <code>role</code>
• <code>user</code>
• <code>roleSubordinates</code>
• <code>roleSubordinatesInternal</code>
• <code>owner</code>
• <code>creator</code>
• <code>partnerUser</code>
• <code>customerPortalUser</code>
• <code>portalRole</code>
• <code>portalRoleSubordinates</code>
• <code>allInternalUsers</code>—all Salesforce users in the organization</td></tr>
</table>

ApprovalPageField

Represents the selection of fields to display on the approval page, where an approver can view the approval request details and approve or reject the record.

Field Name	Field Type	Description
`field`	string[]	An array of fields that are displayed on the page for the approver to approve or reject the record.

ApprovalStep

Represents a step in the approval process. Approval steps define the chain of approval for a particular approval process. Each step determines which records can advance to that step, who to assign approval requests to, and whether to let each approver's delegate respond to the requests. The first step specifies what to do if a record doesn't advance to that step. Later steps specify what happens if an approver rejects the request.

Note:

- The order of the `ApprovalStep` entries in the approval process definition determines the order in which the approval steps are executed.
- After an approval process is activated, you can't add, delete, or change the order of the steps or change its reject or skip behavior, even if the process is inactive.
- Each approval process supports up to 30 steps.

Field Name	Field Type	Description
`allowDelegate`	boolean	Whether to allow delegated approvers in this step of the approval process. A delegated approver is a user appointed by an assigned approver as an alternate for approval requests.
`approvalActions`	ApprovalAction	Specifies which workflow actions to execute when a record is approved in this step of the approval process.
`assignedApprover`	ApprovalStepApprover	Specifies the assigned approvers for this step of the approval process.
`description`	string	Describes the approval step.
`entryCriteria`	ApprovalEntryCriteria	Determines which records can enter this step of the approval process.
`ifCriteriaNotMet`	StepCriteriaNotMetType (enumeration of type string)	Specifies what to do for records that don't meet the entry criteria. Valid values are: • `ApproveRecord`—Approve the request and execute all final approval actions. • `RejectRecord`—Reject the request and execute all final rejection actions. This option is available only for the first step in the approval process. • `GotoNextStep`—Skip to the next approval step. If you select this option for the first approval step, and a record doesn't meet the entry criteria for any other step, the record is rejected.
`label`	string	Required. Name of the approval step.

Field Name	Field Type	Description
`name`	string	Required. Unique name of the approval step. It must contain only underscores and alphanumeric characters, begin with a letter, not include spaces, not end with an underscore, and not contain two consecutive underscores. The requirement for uniqueness is only within the specific approval process.
`rejectBehavior`	ApprovalStepRejectBehavior	Required, except for the first step in the approval process. Specifies what happens if the approver rejects the request during this approval step, unless it's the first step in the approval process. If the approver rejects the request in the first step in the approval process, the reject behavior is determined by the `finalRejectionActions`.
`rejectionActions`	ApprovalAction	Specifies which workflow actions to execute when a record is rejected in this step of the approval process.

ApprovalAction

Represents the actions that occur as a result of an approval process.

Field Name	Field Type	Description
`action`	WorkflowActionReference[]	An array of workflow actions to execute.

ApprovalStepApprover

Represents the assigned approvers for an approval step. Each step supports up to 25 approvers.

Field Name	Field Type	Description
`approver`	Approver[]	An array of assigned approvers for this step of the approval process.
`whenMultipleApprovers`	RoutingType (enumeration of type string)	Specifies how to handle approval or rejection when multiple approvers are assigned to the step. Valid values are: • `Unanimous`—(Default) Require unanimous approval from all approvers for this step. The approval request for this step is rejected if any of the approvers reject the request. • `FirstResponse`—Approve or reject based on the first response.

Approver

Represents an assigned approver for an approval step. Check out "Considerations for Setting Approvers" in the Salesforce Help.

Field Name	Field Type	Description
name	string	Identifies an assigned approver. This field is required, except when the type is one of the following and the name is ignored: - adhoc - userHierarchyField
type	NextOwnerType (enumeration of type string)	Combined with the specified name, this identifies an assigned approver. Valid values are: - adhoc—The approver for the step must be selected manually. For the first step, the submitter selects the approver. For the second and later steps, the approver for the previous step selects the approver. For this value, exclude the name field. - user—A user in your organization. For this value, enter a username for the name field. - userHierarchyField—A user specified in a standard or custom user hierarchy field, such as the standard Manager field. For this value, exclude the name field. The user hierarchy field must be defined in the nextAutomatedApprovers for the approval process. - relatedUserField—A user specified in a user lookup field on the submitted record, such as the Last Modified By field. For this value, enter the name of the user lookup field for the name field. - queue—Automatically assign to a queue. For this value, enter the name of the queue for the name field.

ApprovalEntryCriteria

Represents the criteria that records must meet to enter the approval process or an approval step. Specify either filter criteria or a formula, but not both.

Field Name	Field Type	Description
booleanFilter	string	Filter logic for criteriaItems. Exclude this field if you enter a formula.
criteriaItems	FilterItem[]	Filter criteria that a record must meet to enter the approval process or approval step. Approval processes don't support valueField entries in filter criteria.
formula	string	Formula that must evaluate to true for a record to enter the approval process or approval step.

ApprovalStepRejectBehavior

Represents what happens if the approver rejects the request during this approval step, unless it's the first step in the approval process. For the first step in the approval process, the reject behavior is determined by the approval process's final rejection actions.

Field Name	Field Type	Description
`type`	StepRejectBehaviorType (enumeration of type string)	Not allowed in the first step of the approval process. Valid values are: • `RejectRequest`—Rejects the request even if previous steps were approved. Salesforce performs all rejection actions specified for this step and all final rejection actions. • `BackToPrevious`—Rejects the request, and returns the approval request to the previous approver. Salesforce performs all rejection actions specified for this step.

NextAutomatedApprover

Represents the user hierarchy field to use as the next automated approver for the approval process. If defined, the user specified in the hierarchy field can be automatically assigned as the approver in one or more approval steps.

Field Name	Field Type	Description
`useApproverFieldOfRecordOwner`	boolean	Required. Whether the first executed approval step should use the specified `userHierarchyField` in the record owner's user record—instead of the submitter's user record—as the approver. All remaining steps use the specified `userHierarchyField` in the user record of the preceding step's approver.
`userHierarchyField`	string	Required. Standard or custom user hierarchy field whose value specifies which user to assign as the approver. For example, the standard `Manager` hierarchy field can be used to assign approvers for employee PTO (paid time off) requests.

Declarative Metadata Sample Definition

The following is an example of an ApprovalProcess component:

```xml
<?xml version="1.0" encoding="UTF-8"?>
<ApprovalProcess xmlns="http://soap.sforce.com/2006/04/metadata">
    <active>false</active>
    <allowRecall>false</allowRecall>
    <allowedSubmitters>
        <type>owner</type>
    </allowedSubmitters>
    <allowedSubmitters>
        <submitter>USSalesRep</submitter>
        <type>role</type>
    </allowedSubmitters>
    <allowedSubmitters>
        <submitter>MarketingGroup</submitter>
        <type>group</type>
    </allowedSubmitters>
    <allowedSubmitters>
```

```
        <submitter>kcooper@example.com</submitter>
        <type>user</type>
    </allowedSubmitters>
    <approvalPageFields>
        <field>Name</field>
        <field>Owner</field>
        <field>MyLeadCustomField__c</field>
        <field>Address</field>
    </approvalPageFields>
    <approvalStep>
        <allowDelegate>false</allowDelegate>
        <approvalActions>
            <action>
                <name>LeadApprovedTask1</name>
                <type>Task</type>
            </action>
            <action>
                <name>LeadApprovedTask2</name>
                <type>Task</type>
            </action>
        </approvalActions>
        <assignedApprover>
            <approver>
                <type>adhoc</type>
            </approver>
        </assignedApprover>
        <label>Step1</label>
        <name>Step1</name>
        <rejectionActions>
            <action>
                <name>LeadRejectedTask</name>
                <type>Task</type>
            </action>
        </rejectionActions>
    </approvalStep>
    <approvalStep>
        <allowDelegate>false</allowDelegate>
        <assignedApprover>
            <approver>
                <type>userHierarchyField</type>
            </approver>
        </assignedApprover>
        <entryCriteria>
            <criteriaItems>
                <field>Lead.CreatedDate</field>
                <operation>greaterThan</operation>
                <value>3/25/2013</value>
            </criteriaItems>
            <criteriaItems>
                <field>User.IsActive</field>
                <operation>notEqual</operation>
                <value>true</value>
            </criteriaItems>
        </entryCriteria>
```

```
            <ifCriteriaNotMet>ApproveRecord</ifCriteriaNotMet>
            <label>Step2</label>
            <name>Step2</name>
            <rejectBehavior>
                <type>RejectRequest</type>
            </rejectBehavior>
        </approvalStep>
        <approvalStep>
            <allowDelegate>true</allowDelegate>
            <assignedApprover>
                <approver>
                    <name>MarketingTeamQueue</name>
                    <type>queue</type>
                </approver>
                <approver>
                    <name>LastModifiedBy</name>
                    <type>relatedUserField</type>
                </approver>
                <approver>
                    <name>awheeler@example.com</name>
                    <type>user</type>
                </approver>
                <whenMultipleApprovers>FirstResponse</whenMultipleApprovers>
            </assignedApprover>
            <entryCriteria>
                <formula>CONTAINS( MyLeadCustomField__c , 'Salesforce')</formula>
            </entryCriteria>
            <label>Step3</label>
            <name>Step3</name>
            <rejectBehavior>
                <type>BackToPrevious</type>
            </rejectBehavior>
        </approvalStep>
        <emailTemplate>MyFolder/LeadsNewassignmentnotification</emailTemplate>
        <enableMobileDeviceAccess>false</enableMobileDeviceAccess>
        <entryCriteria>
            <criteriaItems>
                <field>Lead.AnnualRevenue</field>
                <operation>greaterThan</operation>
                <value>10500</value>
            </criteriaItems>
            <criteriaItems>
                <field>Lead.MyLeadCustomField__c</field>
                <operation>equals</operation>
                <value>Salesforce</value>
            </criteriaItems>
        </entryCriteria>
        <finalApprovalActions>
            <action>
                <name>LeadEmailContacted</name>
                <type>Alert</type>
            </action>
        </finalApprovalActions>
        <finalApprovalRecordLock>true</finalApprovalRecordLock>
```

```
    <finalRejectionActions>
        <action>
            <name>ProcessRejectedMessageAction</name>
            <type>OutboundMessage</type>
        </action>
    </finalRejectionActions>
    <finalRejectionRecordLock>false</finalRejectionRecordLock>
    <initialSubmissionActions>
        <action>
            <name>LeadFieldUpdate</name>
            <type>FieldUpdate</type>
        </action>
        <action>
            <name>NewLeadEmail</name>
            <type>Alert</type>
        </action>
    </initialSubmissionActions>
    <label>SampleProcess</label>
    <nextAutomatedApprover>
        <useApproverFieldOfRecordOwner>false</useApproverFieldOfRecordOwner>
        <userHierarchyField>customlookupuserfield__c</userHierarchyField>
    </nextAutomatedApprover>
    <postTemplate>MyPostTemplate</postTemplate>
    <recallActions>
        <action>
            <name>ProcessRecalledMessageAction</name>
            <type>OutboundMessage</type>
        </action>
    </recallActions>
    <recordEditability>AdminOnly</recordEditability>
    <showApprovalHistory>false</showApprovalHistory>
</ApprovalProcess>
```

Wildcard Support in the Manifest File

Use the wildcard character * (asterisk) in the package.xml manifest file to retrieve all approval processes for all objects. You can't use it to retrieve a subset of approval processes. Syntax such as Lead.* is not supported. For information about using the manifest file, see Deploying and Retrieving Metadata with the Zip File.

AssignmentRules

Represents assignment rules that allow you to automatically route cases to the appropriate users or queues. You can access rules metadata for all applicable objects, for a specific object, or for a specific rule on a specific object.

The package.xml syntax for accessing all assignment rules for all objects is:

```
    <types>
        <members>*</members>
        <name>AssignmentRules</name>
    </types>
```

All rules for a specific object uses a similar syntax without the wildcard. For example, all assignment rules for the Case object would use this syntax:

```
<types>
    <members>Case</members>
    <name>AssignmentRules</name>
</types>
```

You can also access specific assignment rules for an object. The following example only accesses the “samplerule” and “newrule” assignment rules on the Case object. Notice that for this example the type name syntax is `AssignmentRule` and not `AssignmentRules`.

```
<types>
    <members>Case.samplerule</members>
    <members>Case.newrule</members>
    <name>AssignmentRule</name>
</types>
```

File Suffix and Directory Location

Assignment rules for an object have the suffix `.assignmentRules` and are stored in the `assignmentRules` folder. For example, all Case assignment rules are stored in the `Case.assignmentRules` file.

Version

AssignmentRules components are available in API version 27.0 and later.

Fields

Field Name	Field Type	Description
`assignmentRule`	AssignmentRule[]	Represents the definitions of the named assignment rules.

AssignmentRule

Specifies whether the rule is active or not and its definition. Rules are processed in the order they appear within the AssignmentRules container.

Field Name	Field Type	Description
`active`	boolean	Indicates whether the assignment rule is active (`true`) or not (`false`).
`fullname`	string	Inherited from Metadata, this field is not defined in the WSDL for this metadata type. It must be specified when creating, updating, or deleting. See `createMetadata()` to see an example of this field specified for a call. This value cannot be `null`.

Field Name	Field Type	Description
ruleEntry	RuleEntry[]	Represents the type and description for the assignment rule.

RuleEntry

Represents the fields used by the rule.

Field Name	Field Type	Description
assignedTo	string	The name of the user or queue the item is assigned to.
assignedToType	AssignToLookupValueType (enumeration of type string)	Valid values are: • User • Queue
booleanFilter	string	Advanced filter conditions that were specified for the rule.
criteriaItems	FilterItem[]	The items in the list that define the assignment criteria.
formula	string	The validation formula. Note: Specify either formula or criteriaItems, but not both fields.
notifyCcRecipients	boolean	Specifies whether email addresses included on the Cc line of an incoming Email-to-Case or Web-to-Lead message should be included on the Cc line of the auto-response to that message (true) or not (false). Available in API version 32.0 and later.
overrideExistingTeams	boolean	Specifies whether the case team should be reset when the assignment is done true) or if the current team is added to the case instead of replacing the previous team (false).
team	string[]	The name of the case team. It may occur 0 or more times.
template	string	Specifies the template to use for the email that is automatically sent to the designated recipient. Lightning email templates aren't packageable. We recommend using a Classic email template.

Declarative Metadata Sample Definition

The following is an example file showing two assignment rules on the Case object:

```
<AssignmentRules xmlns="http://soap.sforce.com/2006/04/metadata"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
    <assignmentRule>
```

```
        <fullName>samplerule</fullName>
        <active>false</active>
        <ruleEntry>
            <assignedTo>testUser@org.com</assignedTo>
            <assignedToType>User</assignedToType>
            <criteriaItems>
                <field>Case.IsEscalated</field>
                <operation>equals</operation>
                <value>True</value>
            </criteriaItems>
            <template>emailtemplate</template>
        </ruleEntry>
    </assignmentRule>
    <assignmentRule>
        <fullName>Another samplerule</fullName>
        <active>false</active>
        <ruleEntry>
            <assignedTo>otherUser@org.com</assignedTo>
            <assignedToType>User</assignedToType>
            <criteriaItems>
                <field>Case.IsEscalated</field>
                <operation>equals</operation>
                <value>False</value>
            </criteriaItems>
            <template>emailtemplate</template>
        </ruleEntry>
    </assignmentRule>
</AssignmentRules>
```

Wildcard Support in the Manifest File

This metadata type supports the wildcard character `*` (asterisk) in the `package.xml` manifest file. For information about using the manifest file, see Deploying and Retrieving Metadata with the Zip File.

AssistantContextItem (Beta)

AssistantContextItem is for internal use only.

Note: As a beta feature, Einstein Voice Assistant is a preview and isn't part of the "Services" under your master subscription agreement with Salesforce. Use this feature at your sole discretion, and make your purchase decisions only on the basis of generally available products and features. Salesforce doesn't guarantee general availability of this feature within any particular time frame or at all, and we can discontinue it at any time. This feature is for evaluation purposes only, not for production use. It's offered as is, and Salesforce has no liability for any harm or damage arising out of or in connection with it. All restrictions, Salesforce reservation of rights, obligations concerning the Services, and terms for related Non-Salesforce Applications and Content apply equally to your use of this feature. For information on enabling this feature in your org, contact your Account Executive. To provide feedback on Einstein Voice Assistant, contact Salesforce Customer Support.

AssistantDefinition (Beta)

AssistantDefinition and its child AssistantDefinitionProfile are for internal use only.

Note: As a beta feature, Einstein Voice Assistant is a preview and isn't part of the "Services" under your master subscription agreement with Salesforce. Use this feature at your sole discretion, and make your purchase decisions only on the basis of generally available products and features. Salesforce doesn't guarantee general availability of this feature within any particular time frame or at all, and we can discontinue it at any time. This feature is for evaluation purposes only, not for production use. It's offered as is, and Salesforce has no liability for any harm or damage arising out of or in connection with it. All restrictions, Salesforce reservation of rights, obligations concerning the Services, and terms for related Non-Salesforce Applications and Content apply equally to your use of this feature. For information on enabling this feature in your org, contact your Account Executive. To provide feedback on Einstein Voice Assistant, contact Salesforce Customer Support.

AssistantSkillQuickAction (Beta)

AssistantSkillQuickAction and its child AssistantSkillQuickActionParam are for internal use only.

Note: As a beta feature, Einstein Voice Assistant is a preview and isn't part of the "Services" under your master subscription agreement with Salesforce. Use this feature at your sole discretion, and make your purchase decisions only on the basis of generally available products and features. Salesforce doesn't guarantee general availability of this feature within any particular time frame or at all, and we can discontinue it at any time. This feature is for evaluation purposes only, not for production use. It's offered as is, and Salesforce has no liability for any harm or damage arising out of or in connection with it. All restrictions, Salesforce reservation of rights, obligations concerning the Services, and terms for related Non-Salesforce Applications and Content apply equally to your use of this feature. For information on enabling this feature in your org, contact your Account Executive. To provide feedback on Einstein Voice Assistant, contact Salesforce Customer Support.

AssistantSkillSobjectAction (Beta)

AssistantSkillSobjectAction and its child AssistantSkillSobjectParam are for internal use only.

Note: As a beta feature, Einstein Voice Assistant is a preview and isn't part of the "Services" under your master subscription agreement with Salesforce. Use this feature at your sole discretion, and make your purchase decisions only on the basis of generally available products and features. Salesforce doesn't guarantee general availability of this feature within any particular time frame or at all, and we can discontinue it at any time. This feature is for evaluation purposes only, not for production use. It's offered as is, and Salesforce has no liability for any harm or damage arising out of or in connection with it. All restrictions, Salesforce reservation of rights, obligations concerning the Services, and terms for related Non-Salesforce Applications and Content apply equally to your use of this feature. For information on enabling this feature in your org, contact your Account Executive. To provide feedback on Einstein Voice Assistant, contact Salesforce Customer Support.

AssistantVersion (Beta)

AssistantVersion and its children AssistantSkill, AssistantSkillIntent, and AssistantVersionAction are for internal use only.

Note: As a beta feature, Einstein Voice Assistant is a preview and isn't part of the "Services" under your master subscription agreement with Salesforce. Use this feature at your sole discretion, and make your purchase decisions only on the basis of generally available products and features. Salesforce doesn't guarantee general availability of this feature within any particular time frame or at all, and we can discontinue it at any time. This feature is for evaluation purposes only, not for production use. It's offered as is, and Salesforce has no liability for any harm or damage arising out of or in connection with it. All restrictions, Salesforce reservation of rights, obligations concerning the Services, and terms for related Non-Salesforce Applications and Content apply equally to your use of this feature. For information on enabling this feature in your org, contact your Account Executive. To provide feedback on Einstein Voice Assistant, contact Salesforce Customer Support.

Audience

Represents the audience in a community. An audience consists of different types of criteria, where the audience can be assigned and used for targeting in a community. This type extends the Metadata metadata type and inherits its `fullName` field.

File Suffix and Directory Location

Audience components have the suffix `.audience` and are stored in the `audience` folder.

Version

Audience components are available in API version 44.0 and later.

Special Access Rules

Access to the Audience type requires the AudienceMetadata permission. This permission is on by default for orgs that have Networks enabled.

Access to permission criteria for the Audience type requires the AudiencePermissionCriteria permission. This permission is available in API version 45.0 and later and is on by default for orgs that have Networks enabled.

Fields

Field Name	Field Type	Description
`audienceName`	string	Required. The name of the audience.
`container`	string	Required. The name of the community or org that contains the audience.
`criteria`	AudienceCriteria	Required. Criteria in an audience. This field is available in API version 47.0 and later.
`criterion`	AudienceCriterion[]	Removed. List of criteria in an audience. Warning: This field is available in API version 44.0–46.0. In API version 47.0 and later, use `criteria` instead.
`description`	string	The description of the audience.
`formula`	string	Formula used to determine the audience. This field is available in API version 45.0 and later.
`formulaFilterType`	FormulaFilterType (enumeration of type string)	Indicates the audience's formula type. Valid values are • `AllCriteriaMatch` • `AnyCriterionMatches` • `CustomLogicMatches` (available in API version 45.0 and later)

Field Name	Field Type	Description
isDefaultAudience	boolean	Required. Indicates whether the audience is the default audience (true) or not (false). This field is available in API version 48.0 and later. The default audience file name is of format Default_***Network Name.***audience.
targets	PersonalizationTargetInfos	Targets for the audience. This field is available in API version 47.0 and later.

AudienceCriteria

Represents criteria for an audience. This subtype is available in API version 47.0 and later.

Field Name	Field Type	Description
criterion	AudienceCriterion[]	List of criteria for an audience.

AudienceCriterion

Represents a criterion for an audience.

Field Name	Field Type	Description
criteriaNumber	int	The number associated with the criterion in a formula, for example, (1 AND 2) OR 3. This field is available in API version 45.0 and later.
criterionValue	AudienceCriteriaValue	The value of the criterion.
operator	AudienceCriterion Operator(enumeration of type string)	The operator associated with this criterion. Valid values are: • Equal • NotEqual • GreaterThan • GreaterThanOrEqual • LessThan • LessThanOrEqual • Contains • StartsWith • Includes (available in API version 45.0 and later) • NotIncludes (available in API version 45.0 and later)
type	AudienceCriterion Type(enumeration of type string)	Required. Valid values are: • GeoLocation • Domain • Profile

Field Name	Field Type	Description
		- `FieldBased` - `Permission` (available in API version 45.0 and later) - `Default` (available in API version 47.0 and later) For a list of AudienceCriteriaValue fields that you can use with each AudienceCriterion `type` field value, see this table.

AudienceCriteriaValue

Represents the value of a criterion in an audience. For a list of AudienceCriteriaValue fields that you can use with each AudienceCriterion `type` field value, see this table.

Field Name	Field Type	Description
`city`	string	City of a user. This field can be used only when the value of the AudienceCriterion `type` field is `GeoLocation`.
`country`	string	Country of a user. This field can be used only when the value of the AudienceCriterion `type` field is `GeoLocation`.
`domain`	string	Domain of a user. This field can be used only when the value of the AudienceCriterion `type` field is `Domain`.
`entityField`	string	Field of an object. This field can be used only when the value of the AudienceCriterion `type` field is `FieldBased`.
`entityType`	string	Type of object. This field can be used only when the value of the AudienceCriterion `type` field is `FieldBased`.
`fieldValue`	string	Value of a field. This field can be used only when the value of the AudienceCriterion `type` field is `FieldBased`.
`isEnabled`	string	Indicates whether the permission is enabled (`true`) or not (`false`) for a user. This field is available in API version 45.0 and later and can be used only when the value of the AudienceCriterion `type` field is `Permission`.
`permissionName`	string	Valid API name of a standard user or custom permission. This field is available in API version 45.0 and later and can be used only when the value of the AudienceCriterion `type` field is `Permission`.
`permissionType`	string	Type of permission. Valid values are `Standard` and `Custom`. This field is available in API version 45.0 and later and can be used only when the value of the AudienceCriterion `type` field is `Permission`.
`profile`	string	Profile of a user. This field can be used only when the value of the AudienceCriterion `type` field is `Profile`.
`subdivision`	string	Subdivision of a user. This field can be used only when the value of the AudienceCriterion `type` field is `GeoLocation`.

This table summarizes which AudienceCriteriaValue fields you can use with the different AudienceCriterion `type` field values.

AudienceCriterion Type	AudienceCriteriaValue Fields

`GeoLocation`	`city`
	`country`
	`subdivision`

`Domain`	`domain`
`Profile`	`profile`

`FieldBased`	`entityField`
	`entityType`
	`fieldValue`

`Permission`	`isEnabled`
	`permissionName`
	`permissionType`

PersonalizationTargetInfos

Represents targets for an audience. This subtype is available in API version 47.0 and later.

Field Name	Field Type	Description
`target`	PersonalizationTargetInfo[]	List of targets for an audience.

PersonalizationTargetInfo

Represents a target for an audience. This subtype is available in API version 47.0 and later.

Field Name	Field Type	Description
`groupName`	string	Required. Group name of the target. Groups bundle related target and audience pairs. Determining group names for targets in Metadata API works the same way as in Chatter REST API. To determine the `groupName` for targets, see Personalization Target Group Names in *Chatter REST API Developer Guide*.
`priority`	int	Priority of the target. Within a group, priority determines which target is returned if the user matches more than one audience.

Field Name	Field Type	Description
targetType	string	Required. Type of target, indicating the nature of the data being targeted. In API version 47.0 and later, ExperienceVariation is the only supported value. For more information on the ExperienceVariation target type, see ExperienceBundle.
targetValue	string	Required. Value of the target, which is the developer name of the experience variation, such ContactSupport_ContactSupportFor California_Page for a page variation.

Declarative Metadata Sample Definition

The following is an example of an Audience component.

```
<?xml version="1.0" encoding="UTF-8"?>
<Audience xmlns="http://soap.sforce.com/2006/04/metadata">
    <audienceName>Audience Metadata</audienceName>
    <container>Customer</container>
    <criteria>
        <criterion>
            <criteriaNumber>1</criteriaNumber>
            <criterionValue>
                <country>United States</country>
                <subdivision>Nevada</subdivision>
            </criterionValue>
            <operator>Equal</operator>
            <type>GeoLocation</type>
        </criterion>
        <criterion>
            <criteriaNumber>2</criteriaNumber>
            <criterionValue>
                <profile>customer community user</profile>
            </criterionValue>
            <operator>Equal</operator>
            <type>Profile</type>
        </criterion>
        <criterion>
            <criteriaNumber>3</criteriaNumber>
            <criterionValue>
                <domain>sampledomain.example.com</domain>
            </criterionValue>
            <operator>Equal</operator>
            <type>Domain</type>
        </criterion>
        <criterion>
            <criteriaNumber>4</criteriaNumber>
            <criterionValue>
              <entityField>Manager.Profile.CreatedBy.Contact.MailingCountry</entityField>

                <entityType>User</entityType>
                <fieldValue>USA</fieldValue>
            </criterionValue>
            <operator>StartsWith</operator>
            <type>FieldBased</type>
        </criterion>
        <criterion>
            <criteriaNumber>5</criteriaNumber>
            <criterionValue>
                <entityField>RecordTypeId</entityField>
                <entityType>CollaborationGroup</entityType>
                <fieldValue>CollaborationGroup.Group_RT2</fieldValue>
            </criterionValue>
            <operator>Equal</operator>
            <type>FieldBased</type>
        </criterion>
        <criterion>
            <criteriaNumber>6</criteriaNumber>
            <criterionValue>
                <isEnabled>true</isEnabled>
                <permissionName>ManageUsers</permissionName>
                <permissionType>Standard</permissionType>
            </criterionValue>
            <operator>Equal</operator>
            <type>Permission</type>
        </criterion>
        <criterion>
            <criteriaNumber>7</criteriaNumber>
            <criterionValue>
                <isEnabled>false</isEnabled>
                <permissionName>NamespaceXYZ__CustomPermABC</permissionName>
                <permissionType>Custom</permissionType>
            </criterionValue>
            <operator>Equal</operator>
            <type>Permission</type>
        </criterion>
    </criteria>
    <formula>1 AND (2 OR 3 OR 4 OR 5 OR 6 OR 7)</formula>
    <formulaFilterType>CustomLogicMatches</formulaFilterType>
    <isDefaultAudience>false</isDefaultAudience>
    <targets>
        <target>
            <groupName>c194d79c-5c6b-4c6a-8d14-0e7042564355$#$Branding</groupName>
            <priority>1</priority>
            <targetType>ExperienceVariation</targetType>
            <targetValue>Customer_Service_testBrandingSet_Branding</targetValue>
        </target>
    </targets>
</Audience>
```

Usage

You can't delete an audience using Metadata API.

In API version 47.0 and later, you can't create an audience without criteria.

The list of targets provided in the input for an audience is considered the state of target assignments that you want. For example, see the following information for deleting, creating, and updating targets.

Note: If you don't have a default audience, updating targets can result in the UI erroneously showing a target assigned to the default audience. The target assignment data in the API is correct. To work around the UI issue, temporarily assign another target to the default audience and then delete it.

Delete targets

To delete a single target from an audience, deploy the entire list of targets for the audience minus the one you want to delete.

To delete all the targets from an audience, deploy the audience with empty targets tags. For example:

```
<?xml version="1.0" encoding="UTF-8"?>
<Audience
    xmlns="http://soap.sforce.com/2006/04/metadata">
    <audienceName>testAudience</audienceName>
    <container>testContainer</container>
    <criteria>
        <criterion>
            <criteriaNumber>1</criteriaNumber>
            <criterionValue>
                <country>United States</country>
                <subdivision>Nevada</subdivision>
            </criterionValue>
            <operator>Equal</operator>
            <type>GeoLocation</type>
        </criterion>
    </criteria>
    <formulaFilterType>AllCriteriaMatch</formulaFilterType>
    <isDefaultAudience>false</isDefaultAudience>
    <targets>
    </targets>
</Audience>
```

Update an audience without updating targets

To update an audience without updating targets, deploy the audience without targets tags. For example:

```
<?xml version="1.0" encoding="UTF-8"?>
<Audience
    xmlns="http://soap.sforce.com/2006/04/metadata">
    <audienceName>testAudience</audienceName>
    <container>testContainer</container>
    <criteria>
        <criterion>
            <criteriaNumber>1</criteriaNumber>
            <criterionValue>
                <country>United States</country>
                <subdivision>Nevada</subdivision>
            </criterionValue>
            <operator>Equal</operator>
            <type>GeoLocation</type>
        </criterion>
    </criteria>
    <formulaFilterType>AllCriteriaMatch</formulaFilterType>
    <isDefaultAudience>false</isDefaultAudience>
</Audience>
```

Create targets

To create a target, deploy the entire list of targets for the audience plus the one you want to create.

Update the priority of a target

To change the priority of a target within an audience, deploy the entire list of targets for the audience with the new priority values for the targets.

To change the priority of a target that affects priority in another audience, deploy both audiences with their entire list of targets with the new priority values for the targets.

Update the target assignment for an audience

To reassign a target to a new audience, deploy both audiences with their entire list of targets; one list with the target removed, and the other list with the target added.

Wildcard Support in the Manifest File

This metadata type supports the wildcard character `*` (asterisk) in the `package.xml` manifest file. For information about using the manifest file, see Deploying and Retrieving Metadata with the Zip File.

AuraDefinitionBundle

Represents an Aura definition bundle. A bundle contains an Aura definition, such as an Aura component, and its related resources, such as a JavaScript controller. The definition can be a component, application, event, interface, or a tokens collection.

File Suffix and Directory Location

An AuraDefinitionBundle component is a folder that contains definition files. Unlike most other metadata components, an AuraDefinitionBundle component isn't a single file, it's a collection of files. Each file represents a resource in a bundle, such as markup, applications, code files (including controllers and helpers), events, documentation, and interfaces. For example, this directory structure shows the hierarchy of the folders and files for two bundles: `bundle1` and `bundle2`.

```
aura
    bundle1
        bundle1.cmp
        bundle1Controller.js
    bundle2
        bundle2.app
        bundle2Controller.js
        bundle2.auradoc
```

Aura definition bundles must be under a top-level folder named `aura`. Each bundle must have its own subfolder under the `aura` folder. The name of each definition file must start with the bundle name.

A bundle doesn't have a suffix. Definition files can have one of these suffixes:

Suffix	Component Type
`.app`	Application
`.cmp`	Component
`.design`	Design

Suffix	Component Type
`.evt`	Event
`.intf`	Interface
`.js`	Controller, Helper, or Renderer
`.svg`	SVG image
`.css`	Style
`.auradoc`	Documentation
`.tokens`	Tokens collection

Each bundle can have only one file each with a suffix of `.app`, `.cmp`, `.design`, `.evt`, `.intf`, or `.tokens`.

Version

AuraDefinitionBundle components are available in API version 32.0 and later.

Design and SVG components are available in API version 33.0 and later.

In API version 45.0 and later, there are two types of Lightning component: Aura components and Lightning web components. This metadata type describes an Aura component.

Special Access Rules

Definitions can be created only in organizations with defined namespaces.

Fields

Field Name	Field Type	Description
`apiVersion`	double	The API version for this definition bundle. When you create an Aura bundle, you can specify the API version to save it with. Available in API version 35.0 and later.
`controllerContent`	base64Binary	The content of a JavaScript client-side controller.
`description`	string	The specification of the Aura bundle. Available in API version 35.0 and later.
`designContent`	base64Binary	The content of a design definition. Only valid inside a component bundle.
`documentationContent`	base64Binary	The content of a documentation definition.
`helperContent`	base64Binary	The content of a JavaScript helper.
`markup`	base64Binary	The content of the markup for a definition.
`modelContent`	base64Binary	Deprecated. Do not use.

Field Name	Field Type	Description
packageVersions	PackageVersion[]	The list of installed managed package versions that this Aura definition bundle references. Available in API version 35.0 and later.
rendererContent	base64Binary	The content of a JavaScript client-side renderer.
styleContent	base64Binary	The CSS for the definition.
SVGContent	base64Binary	The SVG image for the definition.
testsuiteContent	base64Binary	Reserved for future use.
type	AuraBundleType (enumeration of type string)	The definition type. Valid values are: • Application • Component • Event • Interface • Tokens

Declarative Metadata Sample Definition

This example shows the directory structure of an AuraDefinitionBundle component.

```
aura
    sampleCmp
        sampleCmp.cmp
        sampleCmpController.js
```

The following samples show the contents of the metadata definition files that correspond to the sample `aura` directory.

Content of `sampleCmp.cmp`:

```
<aura:component>
    <aura:attribute name="val1" type="String" default="Value"/>
    <aura:attribute name="val2" type="String" />
    <aura:handler name="init" value="{!this}" action="{!c.myAction}"/>
        <ui:outputText value='Hello world!'/>
        <ui:outputText value='{!v.val1}'/>
        <ui:outputText value='{!v.val2}'/>
</aura:component>
```

Content of `sampleCmpController.js`:

```
({
  myAction : function(component) {
    component.set('v.val1','Value1');
    component.set('v.val2','Value2');
  }
})
```

This `package.xml` references the definitions of all Lightning components that are present in the `sampleCmp` bundle.

```
<?xml version="1.0" encoding="UTF-8"?>
<Package xmlns="http://soap.sforce.com/2006/04/metadata">
    <types>
        <members>sampleCmp</members>
        <name>AuraDefinitionBundle</name>
    </types>
    <version>48.0</version>
</Package>
```

Wildcard Support in the Manifest File

This metadata type supports the wildcard character `*` (asterisk) in the `package.xml` manifest file. For information about using the manifest file, see Deploying and Retrieving Metadata with the Zip File.

AuthProvider

Represents an authentication provider (auth provider). An auth provider lets users log in to your Salesforce org from an external service provider, such as Facebook, Google, or GitHub. This type extends the Metadata metadata type and inherits its `fullName` field.

File Suffix and Directory Location

Authentication providers are stored in the `authproviders` directory. The file name matches the URL suffix, and the extension is `.authprovider`. For example, an auth provider with URL suffix `FacebookProvider` is stored in `authproviders/FacebookProvider.authprovider`.

Version

Authentication providers are available in API version 27.0 and later.

Special Access Rules

Only users with the Customize Application and Manage AuthProviders permissions can access this object.

Fields

Field Name	Field Type	Description
`authorizeUrl`	string	Required when creating an OpenID Connect authentication provider. The OAuth authorization endpoint URL. Available in API version 29.0 and later. In API version 33.0 and later, for Salesforce-managed auth providers, leave the field blank to let Salesforce supply and manage the value. For details, see Usage.

Field Name	Field Type	Description
consumerKey	string	The app's key that is registered at the third-party (external) authentication provider. In API version 33.0 and later, for Salesforce-managed auth providers, leave the field blank to let Salesforce supply and manage the value. For details, see Usage.
consumerSecret	string	The consumer secret of the app that is registered at the third-party provider. After it's set, you can't change the value. When using `create()`, this field must be encrypted. To create an encrypted form of the consumer secret from plaintext: **1.** Create an authentication provider with the `consumerSecret` plaintext value. **2.** Save the authentication provider. **3.** Create an outbound change set that includes the authentication provider component. The new change set .xml file has an entry in the form `<consumerSecret>++XYZ++</consumerSecret>` where ++XYZ++ is the encrypted secret. In API version 33.0 and later, for Salesforce-managed auth providers, leave the field blank to let Salesforce supply and manage the value. For details, see Usage.
customMetadataTypeRecord	string	Required when creating a custom authentication provider plug-in. The API name of the custom authentication provider. Available in API version 36.0 and later.
defaultScopes	string	For OpenID Connect authentication providers, the scopes to send with the authorization request, if not specified when a flow starts. Available in API version 29.0 and later. In API version 33.0 and later, for Salesforce-managed auth providers, leave the field blank to let Salesforce supply and manage the value. See Usage on page 205.
errorUrl	string	A custom error URL for the authentication provider to use to report errors.
executionUser	string	Required when specifying a registration handler class. The username of the Salesforce admin or system user who runs the Apex handler, which provides the context in which the Apex handler runs. For example, if the Apex handler creates a contact, the creation can be easily traced back to the registration process. In production, use a system user. The user must have the Manage Users permission. Available in API version 27.0 and later.
friendlyName	string	Required. A user-friendly name for the authentication provider.

Field Name	Field Type	Description
iconUrl	string	The path to an icon to use as a button on the login page. Users click the button to log in with the associated authentication provider, such as Twitter or Facebook. Available in API version 32.0 and later.
idTokenIssuer	string	Available when configuring an OpenID Connect authentication provider, the source of the authentication token in https: URI format. If provided, Salesforce validates the returned `id_token` value. OpenID Connect requires returning an `id_token` value with the `access_token` value. Available in API version 30.0 and later.
includeOrgIdInIdentifier	Boolean	Used to differentiate between users with the same user ID from two sources (such as two sandboxes). If enabled (`true`), Salesforce stores the org ID of the third-party identity in addition to the user ID. After you enable this setting, you can't disable it. Applies only to a Salesforce-managed auth provider. Available in API version 32.0 and later.
LinkKickoffUrl	string	The URL for linking existing Salesforce users to a third-party account. This field is read-only. Available in API version 43.0 and later.
logoutUrl	string	The destination for users after they log out if they authenticated using single sign-on. The URL must be fully qualified with an http or https prefix, such as `https://acme.my.salesforce.com`. Available in API version 33.0 and later.
oauthKickoffUrl	string	The URL for obtaining OAuth access tokens for a third party. This field is read-only. Available in API version 43.0 and later.
plugin	string	An existing Apex class that extends the `Auth.AuthProviderPluginClass` abstract class. Available in API version 36.0 and later.
portal	string	This field is used only with portals, which are deprecated. Salesforce doesn't support creating portals, but existing portals are supported.
providerType	AuthProviderType (enumeration of type string)	Required. The third-party authentication provider to use. Valid values include: • `Apple` • `Facebook`. • `Google`. • `Salesforce`. • `Janrain`. • `LinkedIn`. Available in API version 32.0 and later. • `Twitter`. Available in API version 32.0 and later. • `OpenIdConnect`. Available in API version 29.0 and later. • `MicrosoftACS`—Microsoft Access Control Service typically provides authentication for a Microsoft Office 365 service, like SharePoint Online. Available in API version 31.0 and later.

Field Name	Field Type	Description
		• `GitHub`—Provides authentication for a `GitHub` provider. Used to log in users of your Lightning Platform app to GitHub using OAuth. When logged in to GitHub, your app can make calls to GitHub APIs. The `GitHub` provider isn't available as an SSO provider, so users can't log in to your Salesforce org using their GitHub login credentials. Available in API version 35.0 and later. • `Custom`—A provider configured with a custom authentication provider plug-in. Available in API version 36.0 and later.
`registrationHandler`	string	An existing Apex class that implements the `Auth.RegistrationHandler` interface.
`sendAccessTokenInHeader`	boolean	If enabled (`true`), the access token is sent to the `UserInfoUrl` in a header instead of a query string. Available in API version 30.0 and later.
`sendClientCredentialsInHeader`	boolean	Required when creating an OpenID Connect authentication provider. If enabled (`true`), the client credentials are sent in a header to the `tokenUrl` instead of a query string. The credentials are in the standard OpenID Connect Basic Credentials header format, which is `Basic <token>`, where `<token>` is the base64-encoded string `"clientkey:clientsecret"`. Available in API version 30.0 and later.
`sendSecretInApis`	boolean	Determines whether the encrypted consumer secret appears in API responses. If enabled (default), the secret appears in the response. If disabled (`false`), responses don't include the consumer secret. For security, you can disable the setting. However, keep in mind that: • By disabling this setting, the consumer secret is excluded from API responses in all API versions. • Change sets and other metadata deployments break because both the consumer key and secret are expected. To fix this problem, insert the consumer key manually during deployment. Available in API version 47.0 and later.
`SsoKickoffUrl`	string	The URL for performing single sign-on into Salesforce from a third party by using its third-party credentials. This field is read-only. Available in API version 43 and later.
`tokenUrl`	string	The OAuth token endpoint URL of an OpenID Connect authentication provider.Available in API version 29.0 and later. In API version 33.0 and later, for Salesforce-managed auth providers, leave the field blank to let Salesforce supply and manage the value. For details, see Usage.

Field Name	Field Type	Description
userInfoUrl	string	The OpenID Connect endpoint URL of the OpenID Connect authentication provider. Available in API version 29.0 and later. In API version 33.0 and later, for Salesforce-managed auth providers, leave the field blank to let Salesforce supply and manage the value. For details, see Usage.

Declarative Metadata Sample Definition

```
<?xml version="1.0" encoding="UTF-8"?>
<AuthProvider xmlns="http://soap.sforce.com/2006/04/metadata">
    <consumerKey>yourappkey</consumerKey>
    <consumerSecret>PwdVxXjzu3NCZ3MD4He+wA==</consumerSecret>
    <executionUser>admin@your.org</executionUser>
    <friendlyName>FacebookAuthProvider</friendlyName>
    <providerType>Facebook</providerType>
    <registrationHandler>RegistrationHandler</registrationHandler>
    <sendSecretInApis>true</sendSecretInApis>
</AuthProvider>
```

This example package manifest references the previous AuthProvider definition.

```
<?xml version="1.0" encoding="UTF-8"?>
<Package xmlns="http://soap.sforce.com/2006/04/metadata">
    <types>
        <members>FacebookAuthProvider</members>
        <name>AuthProvider</name>
    </types>
    <version>28.0</version>
</Package>
```

Usage

Salesforce provides default authentication providers, called Salesforce-managed auth providers, to simplify setting up these service providers for authentication.

- Apple
- Facebook
- GitHub
- Google
- Janrain
- LinkedIn
- Salesforce
- Twitter

To use a Salesforce-managed auth provider, leave these fields blank when creating your auth provider from the Auth. Provider Setup page.

- authorizeUrl

- `consumerKey`
- `consumerSecret`
- `defaultScopes`
- `tokenURL`
- `userInfoUrl`

Note: If you provide a value for one of these fields, you must also provide a value for `consumerKey` and `consumerSecret`.

Wildcard Support in the Manifest File

This metadata type supports the wildcard character `*` (asterisk) in the `package.xml` manifest file. For information about using the manifest file, see Deploying and Retrieving Metadata with the Zip File.

AutoResponseRules

Represents an auto-response rule that sets conditions for sending automatic email responses to lead or case submissions based on the attributes of the submitted record. You can access rules metadata for all applicable objects, for a specific object, or for a specific rule on a specific object.

The `package.xml` syntax for accessing all auto-response rules for all objects is:

```
<types>
    <members>*</members>
    <name>AutoResponseRules</name>
</types>
```

All rules for a specific object uses a similar syntax without the wildcard. For example, all auto-response rules for the Case object would use this syntax:

```
<types>
    <members>Case</members>
    <name>AutoResponseRules</name>
</types>
```

You can also access specific auto-response rules for an object. The following example only accesses the “samplerule” and “newrule” auto-response rules on the Case object. Notice that for this example the type name syntax is `AutoResponseRule` and not `AutoResponseRules`.

```
<types>
    <members>Case.samplerule</members>
    <members>Case.newrule</members>
    <name>AutoResponseRule</name>
</types>
```

File Suffix and Directory Location

AutoResponseRules for an object have the suffix `.autoResponseRules` and are stored in the `autoResponseRules` folder. For example, all Case auto-response rules are stored in the `Case.autoResponseRules` file.

Version

AutoResponseRules components are available in API version 27.0 and later.

Fields

Field Name	Field Type	Description
`autoresponseRule`	AutoResponseRule[]	Represents the definitions of the named auto-response rules.

AutoResponseRule

Represents whether a rule is active or not and the order in which the entry is processed in the rule.

Field Name	Field Type	Description
`active`	boolean	Indicates whether the autoresponse rule is active (`true`) or not (`false`).
`fullname`	string	Inherited from Metadata, this field is not defined in the WSDL for this metadata type. It must be specified when creating, updating, or deleting. See `createMetadata()` to see an example of this field specified for a call. This value cannot be `null`.
`ruleEntry`	RuleEntry[]	Represents the type and description for the auto-response rule.

RuleEntry

Represents the fields used by the rule.

Field Name	Field Type	Description
`booleanFilter`	string	Advanced filter conditions that were specified for the rule.
`criteriaItems`	`FilterItem[]`	The items in the list that define the assignment criteria.
`formula`	string	The validation formula. Note: Specify either `formula` or `criteriaItems`, but not both fields.
`replyToEmail`	string	The email address that appears in the reply-to header.
`senderEmail`	string	The email address of the person or queue sending the email notification.

Field Name	Field Type	Description
`senderName`	string	The name of the person or queue sending the email notification.
`template`	string	Specifies the template to use for the email that is automatically sent to the designated recipient. Lightning email templates aren't packageable. We recommend using a Classic email template.

Declarative Metadata Sample Definition

The following is an example AutoResponseRules component:

```
<AutoResponseRules xmlns="http://soap.sforce.com/2006/04/metadata">
    <autoResponseRule>
        <fullName>ajbdeploytest2</fullName>
        <active>false</active>
        <ruleEntry>
            <criteriaItems>
                <field>Case.Description</field>
                <operation>contains</operation>
                <value>testing</value>
            </criteriaItems>
            <senderEmail>test@test.org</senderEmail>
            <senderName>tester name j</senderName>
            <replyToEmail>test@@test.org</replyToEmail>
            <template>emailtemplate</template>
        </ruleEntry>
    </autoResponseRule>
</AutoResponseRules>
```

Wildcard Support in the Manifest File

This metadata type supports the wildcard character `*` (asterisk) in the `package.xml` manifest file. For information about using the manifest file, see Deploying and Retrieving Metadata with the Zip File.

Bot

Represents a definition of an Einstein Bot configuration that can have one or more versions. Only one version can be active. This type extends the Metadata metadata type and inherits its `fullName` field.

File Suffix and Directory Location

Bot components have the suffix `.bot` and are stored in the `bot` folder.

Version

Bot components are available in API version 43.0 and later.

Special Access Rules

Bot is available only if Chat and Einstein Bots are enabled in your org.

Fields

Field Name	Field Type	Description
`botMlDomain`	LocalMlDomain	Represents the Einstein intent set that groups intents, entities, and variables associated with a bot. All Einstein Bot versions under the same bot now share an intent set. Available in API version 44.0 and later.
`botUser`	string	Represents a user profile associated with the bot. Available in API version 46.0 and later.
`botVersions`	BotVersion	Represents the configuration details for a specific Einstein Bots version, including dialogs, intents, entities, and variables.
`contextVariables`	ConversationContextVariable	Represents the context variables that enable your bot to gather customer information regardless of channel. Available in API 45.0 and later.
`description`	string	A description of the bot.
`label`	string	Label that identifies the bot throughout the Salesforce user interface.

ConversationContextVariable

A context variable local to the current bot version. Available in API version 45.0 and later.

Field Name	Field Type	Description
`contextVariableMappings`	ConversationContextVariableMapping	Represents the mapping between a context variable, channel type, and sObject field.
`dataType`	ConversationDataType (enumeration of type string)	Required. Represents the data type of the context variable. Required. Valid values are: • `Text` • `Number` • `Boolean` • `Object` • `Date` • `DateTime` • `Currency` • `Id`

Field Name	Field Type	Description
`developerName`	string	Required. Represents the name of the context variable. Required. Can contain only underscores and alphanumeric characters and must be unique in your org. It must begin with a letter, not include spaces, not end with an underscore, and not contain two consecutive underscores.
`label`	string	Required. A label that identifies the context variable throughout the Salesforce user interface.
`SObjectType`	string	Valid values are: - `BotDefinition` - `Queue`

ConversationContextVariableMapping

Represents the mapping between a context variable, channel type, and sObject field.

Field Name	Field Type	Description
`fieldName`	string	Required. The API name of an SObject field to be used as part of the mapping.
`messageType`	MessageType (enumeration of type string)	Required. Represents the channel, either SMS messaging or Chat. Required. Valid values are: - `Text` - `Facebook` - `Line` - `GoogleHome` - `Alexa` - `Omega` - `AppleBusinessChat` - `WeChat` - `WebChat` - `WhatsApp` - `Phone`
`SObjectType`	string	Required. SObject type for the field property defined as part of the mapping. Valid values are: - `LiveChatTranscript` - `MessagingEndUser` - `MessagingSession`

LocalMlDomain

An Einstein Intent Set local to the current bot version.

Field Name	Field Type	Description
label	string	Label that represents an Einstein Intent Set local to the current bot version throughout the Salesforce user interface.
mlIntents	MlIntent[]	List of intents associated with this local intent set.
mlSlotClasses	MlSlotClass[]	List of entities associated with this local intent set.
name	string	Required. This unique name prevents conflicts with other local Einstein Intent Sets. This name can contain only underscores and alphanumeric characters and must be unique in your org. It must begin with a letter, not include spaces, not end with an underscore, and not contain two consecutive underscores.

Declarative Metadata Sample Definition

The following is an example of a Bot. This example has been trimmed to make it easier to read.

```
<?xml version="1.0" encoding="UTF-8"?>
<Bot xmlns="http://soap.sforce.com/2006/04/metadata">
    <botMlDomain>
        <label>Astros Pizza</label>
        <mlIntents>
            <developerName>New_Order</developerName>
            <label>New Order</label>
            <mlIntentUtterances>
                <utterance>Today is pie day so I want pie</utterance>
            </mlIntentUtterances>
        </mlIntents>
        <mlSlotClasses>
            <developerName>Size</developerName>
            <extractionType>Value</extractionType>
            <label>Size</label>
            <mlSlotClassValues>
                <synonymGroup>
                    <languages>en_US</languages>
                    <terms>Big</terms>
                    <terms>Extra Large</terms>
                    <terms>X-Large</terms>
                    <terms>Grande</terms>
                    <terms>Huge</terms>
                </synonymGroup>
                <value>Large</value>
            </mlSlotClassValues>
        </mlSlotClasses>
        <name>Astros_Pizza_ld1</name>
    </botMlDomain>
    <botVersions>
        <fullName>v1</fullName>
        <botDialogGroups>
            <developerName>Order_Management</developerName>
            <label>Order Management</label>
        </botDialogGroups>
        <botDialogs>
            <botDialogGroup>Order_Management</botDialogGroup>
            <botSteps>
                <botMessages>
                    <message>□□□□□□Pizza Time! □□□□□□</message>
                </botMessages>
                <type>Message</type>
            </botSteps>
            <botSteps>
                <botStepConditions>
                    <leftOperandName>Verified_User</leftOperandName>
                    <leftOperandType>ConversationVariable</leftOperandType>
                    <operatorType>Equals</operatorType>
                    <rightOperandValue>false</rightOperandValue>
                </botStepConditions>
                <botSteps>
                    <botNavigation>
                        <botNavigationLinks>
                            <targetBotDialog>Customer_Verification</targetBotDialog>
                        </botNavigationLinks>
                        <type>Call</type>
                    </botNavigation>
                    <type>Navigation</type>
                </botSteps>
                <type>Group</type>
            </botSteps>
            <botSteps>
                <botStepConditions>
                    <leftOperandName>Location</leftOperandName>
                    <leftOperandType>ConversationVariable</leftOperandType>
                    <operatorType>IsNotSet</operatorType>
                </botStepConditions>
                <botSteps>
                    <botNavigation>
                        <botNavigationLinks>
                            <targetBotDialog>Select_Location</targetBotDialog>
                        </botNavigationLinks>
                        <type>Call</type>
                    </botNavigation>
                    <type>Navigation</type>
                </botSteps>
                <type>Group</type>
            </botSteps>
            <botSteps>
                <botVariableOperation>
                    <botInvocation>
                        <invocationActionName>CreateOrderService</invocationActionName>
                        <invocationActionType>apex</invocationActionType>
                        <invocationMappings>
                            <parameterName>customer</parameterName>
                            <type>Input</type>
                            <variableName>Contact</variableName>
                            <variableType>ConversationVariable</variableType>
                        </invocationMappings>
                        <invocationMappings>
                            <parameterName>location</parameterName>
                            <type>Input</type>
                            <variableName>Location</variableName>
                            <variableType>ConversationVariable</variableType>
                        </invocationMappings>
                        <invocationMappings>
                            <parameterName>output</parameterName>
                            <type>Output</type>
                            <variableName>Pizza_Order</variableName>
                            <variableType>ConversationVariable</variableType>
                        </invocationMappings>
                    </botInvocation>
                    <type>Set</type>
                </botVariableOperation>
                <type>VariableOperation</type>
            </botSteps>
            <botSteps>
                <botMessages>
                 <message>Perfect, let&apos;s work on your order from our {!Location.Name}
location</message>
                </botMessages>
                <type>Message</type>
            </botSteps>
            <botSteps>
                <botNavigation>
                    <botNavigationLinks>
                        <targetBotDialog>Add_Items_to_Order</targetBotDialog>
                    </botNavigationLinks>
                    <type>Redirect</type>
                </botNavigation>
                <type>Navigation</type>
            </botSteps>
            <developerName>New_Order</developerName>
            <label>New Order</label>
            <mlIntent>New_Order</mlIntent>
            <showInFooterMenu>false</showInFooterMenu>
        </botDialogs>
        <conversationVariables>
            <dataType>Object</dataType>
            <developerName>Contact</developerName>
            <label>Contact</label>
        </conversationVariables>
        <conversationVariables>
            <dataType>Text</dataType>
            <developerName>Delivery_Address</developerName>
            <label>Delivery Address</label>
        </conversationVariables>
        <conversationVariables>
            <dataType>Object</dataType>
            <developerName>Pizza_Order</developerName>
            <label>Pizza Order</label>
        </conversationVariables>
        <entryDialog>Welcome</entryDialog>
        <mainMenuDialog>Main_Menu</mainMenuDialog>
    </botVersions>
    <contextVariables>
        <contextVariableMappings>
            <SObjectType>LiveChatTranscript</SObjectType>
            <fieldName>LiveChatTranscript.ChatKey</fieldName>
            <messageType>WebChat</messageType>
        </contextVariableMappings>
        <dataType>Text</dataType>
        <developerName>ChatKey</developerName>
        <label>Chat Key</label>
    </contextVariables>
    <contextVariables>
        <contextVariableMappings>
            <SObjectType>LiveChatTranscript</SObjectType>
            <fieldName>LiveChatTranscript.ContactId</fieldName>
            <messageType>WebChat</messageType>
        </contextVariableMappings>
        <dataType>Id</dataType>
        <developerName>ContactId</developerName>
        <label>Contact Id</label>
    </contextVariables>
    <contextVariables>
        <contextVariableMappings>
            <SObjectType>LiveChatTranscript</SObjectType>
            <fieldName>LiveChatTranscript.LiveChatVisitorId</fieldName>
            <messageType>WebChat</messageType>
        </contextVariableMappings>
        <dataType>Id</dataType>
        <developerName>EndUserId</developerName>
        <label>End User Id</label>
    </contextVariables>
    <contextVariables>
        <contextVariableMappings>
            <SObjectType>LiveChatTranscript</SObjectType>
            <fieldName>LiveChatTranscript.Id</fieldName>
            <messageType>WebChat</messageType>
        </contextVariableMappings>
        <dataType>Id</dataType>
        <developerName>RoutableId</developerName>
        <label>Routable Id</label>
    </contextVariables>
    <label>Astro&apos;s Pizza</label>
</Bot>
```

The following is an example `package.xml` that references the previous definition.

```
<?xml version="1.0" encoding="UTF-8"?>
<Package xmlns="http://soap.sforce.com/2006/04/metadata">
    <types>
        <members>Pizza_Bot</members>
        <name>Bot</name>
    </types>
    <version>45.0</version>
</Package>
```

Wildcard Support in the Manifest File

This metadata type supports the wildcard character `*` (asterisk) in the `package.xml` manifest file. For information about using the manifest file, see Deploying and Retrieving Metadata with the Zip File.

BotVersion

Represents the configuration details for a specific Einstein Bot version, including dialogs, intents, entities, and variables. This type extends the Metadata metadata type and inherits its `fullName` field.

File Suffix and Directory Location

BotVersion components have the suffix `.bot` and are stored in the `bot` folder. BotVersion is a top-level child of Bot and shares its suffix and file directory.

Version

BotVersion components are available in API version 43.0 and later.

Special Access Rules

BotVersion is available only if Chat and Einstein Bots are enabled in your org.

Fields

Field Name	Field Type	Description
`botDialogGroups`	BotDialogGroup[]	The list of dialog groups in this bot version.
`botDialogs`	BotDialog[]	The list of dialogs in this bot version.
`conversationVariables`	ConversationVariable[]	A container that stores a specific piece of data collected from the customer. You can use variables within dialog actions as both inputs and outputs. Available in API version 44.0 and later.
`entryDialog`	string	Required. A reference to the first dialog that the bot presents to your customer. For example, `Welcome`.
`mainMenuDialog`	string	Required. A reference to the dialog identified as the main menu dialog. For example, `Main Menu`.
`responseDelayMilliseconds`	int	An optional default or custom delay after every bot response to simulate typing.

BotDialogGroup

The list of dialog groups in this bot version.

Field Name	Field Type	Description
description	string	A description of the bot dialog group.
developerName	string	Required. This unique name prevents conflicts with other dialog groups associated with the same bot version. This name can contain only underscores and alphanumeric characters. The name must begin with a letter, not include spaces, not end with an underscore, and not contain two consecutive underscores.
label	string	Required. A label that identifies the dialog group throughout the Salesforce user interface.

BotDialog

The list of dialogs in this bot version.

Field Name	Field Type	Description
botDialogGroup	string	The bot dialog group that contains this bot dialog.
botSteps	BotStep[]	A list of steps that are executed as part of the dialog.
description	string	A description of the bot dialog.
developerName	string	Required. This unique name prevents conflicts with other dialogs associated with the same bot version. This name can contain only underscores and alphanumeric characters. It must begin with a letter, not include spaces, not end with an underscore, and not contain two consecutive underscores.
label	string	Required. A label that identifies the dialog throughout the Salesforce user interface.
mlIntent	string	Required. The name of the intent associated with a dialog.
mlIntentTrainingEnabled	boolean	Indicates whether Einstein is turned on to train an intent model for the dialog intent (true) or turned off for the exact match option (false). The default value is false.Available in API version 46.0 and later.
showInFooterMenu	boolean	Indicates whether to show this dialog in the Bot Options menu. The default value is false.

BotStep

A step that is executed as part of the dialog.

Field Name	Field Type	Description
booleanFilter	string	This field is reserved for future use.
botInvocation	BotInvocation	Bot Invocation used by a BotStep of type `Invocation`.
botMessages	BotMessage[]	List of bot messages used by a BotStep of type `Message`.
botNavigation	BotNavigation	Bot Navigation used by a BotStep of type `Navigation`.
botStepConditions	BotStepCondition[]	List of BotStep conditions associated with a BotStep of type `Group`.
botSteps	BotStep[]	List of BotSteps associated to a Bot Step of type `Group`.
botVariableOperation	BotVariableOperation[]	Bot Variable Operation used by a BotStep of type `VariableOperation`.
conversationRecordLookup	ConversationRecordLookup[]	A lookup action to the Conversation record. Available in API version 46.0 and later.
conversationSystemMessage	ConversationSystemMessage[]	System messages that represent an action for a Bot Step, such as transferring to an agent or ending a chat. Available in API version 46.0 and later.
type	BotStepType (enumeration of type string)	Required. Valid values are: • `Navigation` • `Invocation` • `VariableOperation` • `Message` • `Wait` • `Group` • `RecordLookup` (Available in API version 48.0 and later.)

BotInvocation

Bot Invocation used by a BotStep of type `Invocation`.

Field Name	Field Type	Description
invocationActionName	string	The name of the invocable action used by a Bot Invocation.
invocationActionType	ConversationInvocableTargetType	Available dialog action types are: • `apex` • `flow` • `standardInvocableAction`
invocationMappings	BotInvocationMapping[]	List of Bot Invocation Mappings for a Bot Invocation.

BotInvocationMapping

List of Bot Invocation Mappings for a Bot Invocation.

Field Name	Field Type	Description
`parameterName`	string	Required. Name of an Input/Output parameter of the parent Bot Invocation target.
`type`	BotInvocationMappingType (enumeration of type string)	Required. Valid values are: - `Input` - `Output`
`value`	string	Literal Value to be assigned to the specified parameter.
`variableName`	string	Name of the Bot Variable that is used as part of an Invocation mapping.
`variableType`	ConversationVariableType (enumeration of type string)	This field relates to the type of variable used in this invocation mapping. Valid values are: - `ConversationVariable` - `ContextVariable`

BotMessage

A bot message used by a BotStep of type `Message`.

Field Name	Field Type	Description
`message`	string	Required. Message to display as part of an outgoing message from the bot to the customer.

BotNavigation

Bot navigation used by a BotStep of type `Navigation`.

Field Name	Field Type	Description
`botNavigationLinks`	BotNavigationLink[]	List of Bot Navigation links associated with a Bot Navigation of type `Call` or `Redirect`.
`type`	BotNavigationType (enumeration of type string)	Required. Valid values are: - `Call` - `Redirect` - `TransferToAgent`

BotNavigationLink

List of Bot Navigation links associated with a Bot Navigation of type `Call` or `Redirect`.

Field Name	Field Type	Description
`label`	string	Label displayed when more than one Bot Navigation Link is available under a Bot Navigation of type `Redirect`. The target dialog label is used when no label is provided.
`targetBotDialog`	string	Required. Name of the target dialog to be called as part of this Bot Navigation Link.

BotStepCondition

List of BotStep conditions associated with a BotStep of type `Group`.

Field Name	Field Type	Description
`leftOperandName`	string	Required. Name of the variable used as the left side of the condition operation.
`leftOperandType`	ConversationVariableType (enumeration of type string)	Required. Type of the variable used as the left side of the condition operation. Valid values are: • `ConversationVariable` • `ContextVariable`
`operatorType`	BotStepConditionOperatorType (enumeration of type string)	Required. Valid values are: • `Equals` • `NotEquals` • `IsSet` • `IsNotSet` • `GreaterThan` (Available in API version 47.0 and later.) • `LessThan` (Available in API version 47.0 and later.) • `GreaterThanOrEqualTo` (Available in API version 47.0 and later.) • `LessThanOrEqualTo` (Available in API version 47.0 and later.)
`rightOperandValue`	string	Value that is used as the right side of the condition operation. This value is ignored when using `IsSet` and `IsNotSet` operators.

BotVariableOperation

Bot variable operation used by a BotStep of type `VariableOperation`.

Field Name	Field Type	Description
`botInvocation`	BotInvocation	Bot Invocation used to provide Dynamic choices by a Bot Variable Operation of type `Collect` and `quickReplyType` of `Dynamic`.
`botMessages`	BotMessage[]	List of Bot Messages used as prompt messages by a Bot Variable Operation of type `Collect`.

Field Name	Field Type	Description
botQuickReplyOptions	BotQuickReplyOption[]	List of static choice options used by a Bot Variable Operation of type Collect and quickReplyType of Static.
botVariableOperands	BotVariableOperand[]	List of Bot Variable Operands associated with a Bot Variable of type Set or Unset.
invalidInputBotNavigation	BotNavigation	Bot Navigation used by a Bot Variable Operation of type Collect. This navigation is executed when the associated Bot Invocation doesn't return any options.
quickReplyOptionTemplate	string	Formula template used to resolve a label for Dynamic choice options of type Object.
quickReplyType	BotQuickReplyType (enumeration of type string)	Valid values are: • Static • Dynamic
quickReplyWidgetType	BotWidgetType (enumeration of type string)	Valid values are: • Menu • Buttons
sourceVariableName	string	Name of the source VariableName used in the variable operation. (Available in API version 47.0 and later.)
sourceVariableType	ConversationVariableType (enumeration of type string)	This name defines the data type of VariableName used in the variable operation. (Available in API version 47.0 and later.) Valid values are: • ConversationVariable • ContextVariable
type	BotVariableOperationType (enumeration of type string)	Required. Valid values are: • Set • Unset • Collect

BotQuickReplyOption

List of static choice options used by a bot variable operation of type Collect and quickReplyType of Static.

Field Name	Field Type	Description
literalValue	string	Required. Value to be displayed as a menu or button choice to your customer.

BotVariableOperand

List of bot variable operands associated with a bot variable of type Set or Unset.

Field Name	Field Type	Description
`disableAutoFill`	boolean	Disables auto-fill behavior for a bot variable under a bot variable operation of type `Collect`.
`sourceName`	string	Name of the source CustomField or MlSlotClass used in the variable operation.
`sourceType`	ConversationVariableOperandSourceType (enumeration of type string)	Valid values are: - `StandardConversationVariable` - `ConversationVariable` - `ContextVariable` Available in API 45.0 and later. - `MlSlotClass` - `StandardMlSlotClass` - `Value`
`sourceValue`	string	Literal value used as the source for this variable operation.
`targetName`	string	Required. Name of the target variable used in the variable operation.
`targetType`	ConversationVariableType (enumeration of type string)	Required. Type of the target used in the variable operation. Valid values are: - `ConversationVariable` - `ContextVariable`

ConversationRecordLookup

Information related to the linked conversation. Currently only works on Lightning Knowledge. Available in API version 46.0 and later.

Field Name	Field Type	Description
`lookupFields`	ConversationRecordLookupField []	Definition of the fields that are used for this lookup.
`maxLookupResults`	int	Required. The maximum numbers of records to return (1-3).
`SObjectType`	string	Required. Specifies the SObjectType of the ID stored in a bot variable.
`sourceVariableName`	string	Required. The input for this lookup operation.
`sourceVariableType`	ConversationVariableType (enumeration of type string)	Required. Type of the target used in the variable operation. Valid values are: - `ConversationVariable` - `ContextVariable`
`targetVariableName`	string	Required. The variable that holds the results of this lookup.

ConversationRecordLookupField

The fields used in a conversation record lookup. Available in API version 46.0 and later.

Field Name	Field Type	Description
`fieldName`	string	Required. Defines the field names used in the Conversation Lookup function.

ConversationSystemMessage

System messages that represent an action for a Bot Step, such as transferring to an agent or ending a chat. Available in API version 46.0 and later.

Field Name	Field Type	Description
`systemMessageMappings`	ConversationSystemMessageMapping	Defines the type of system message to be sent.
`type`	ConversationSystemMessageType (enumeration of type string)	Required. This field defines the values available for a system message. Valid values are: • `Transfer` • `EndChat`

ConversationSystemMessageMapping

List of mappings that indicate additional information provided for the system message. Available in API version 46.0 and later.

Field Name	Field Type	Description
`mappingType`	ConversationMappingType (enumeration of type string)	Required. Defines the type of mapping used in the record. Valid values are `Input` and `Output`.
`parameterType`	ConversationSystemMessageParamType (enumeration of type string)	Required. Defines the type of parameter the value is mapped to. Valid value is `Transfer`.
`variableName`	string	Required. Name of the variable that contains the value passed to the system message.

ConversationVariable

A container that stores a specific piece of data collected from the customer. You can use variables within dialog actions as both inputs and outputs. Available in API version 44.0 and later.

Field Name	Field Type	Description
`collectionType`	ConversationVariableCollectionType (enumeration of type string)	This field defines whether a variable is designated as a List Variable. Valid value is `List`.

Field Name	Field Type	Description
`dataType`	ConversationVariableDataType (enumeration of type string)	Required. Valid values are: - `Boolean` - `Currency` - `Date` - `DateTime` - `Id` (available in API 45.0 and later.) - `Object` - `Number` - `Text`
`developerName`	string	Required. This name can contain only underscores and alphanumeric characters and must be unique in your org. It must begin with a letter, not include spaces, not end with an underscore, and not contain two consecutive underscores.
`label`	string	Required. Label that identifies a variable throughout the Salesforce user interface.
`SObjectType`	string	Specifies the SObjectType of the ID stored in a bot variable. Valid values are: - `BotDefinition` - `Queue`

Declarative Metadata Sample Definition

The following is an example of a BotVersion.

```
<?xml version="1.0" encoding="UTF-8"?>
<Bot xmlns="http://soap.sforce.com/2006/04/metadata">
    <botMlDomain>
        <label>Astros Pizza</label>
        <mlIntents>
            <developerName>New_Order</developerName>
            <label>New Order</label>
            <mlIntentUtterances>
                <utterance>Today is pie day so I want pie</utterance>
            </mlIntentUtterances>
        </mlIntents>
        <mlSlotClasses>
            <developerName>Size</developerName>
            <extractionType>Value</extractionType>
            <label>Size</label>
            <mlSlotClassValues>
                <synonymGroup>
                    <languages>en_US</languages>
                    <terms>Big</terms>
                    <terms>Extra Large</terms>
                    <terms>X-Large</terms>
                    <terms>Grande</terms>
                    <terms>Huge</terms>
                </synonymGroup>
                <value>Large</value>
            </mlSlotClassValues>
        </mlSlotClasses>
        <name>Astros_Pizza_ld1</name>
    </botMlDomain>
    <botVersions>
        <fullName>v1</fullName>
        <botDialogGroups>
            <developerName>Order_Management</developerName>
            <label>Order Management</label>
        </botDialogGroups>
        <botDialogs>
            <botDialogGroup>Order_Management</botDialogGroup>
            <botSteps>
                <botMessages>
                    <message>□□□□□□Pizza Time! □□□□□□</message>
                </botMessages>
                <type>Message</type>
            </botSteps>
            <botSteps>
                <botStepConditions>
                    <leftOperandName>Verified_User</leftOperandName>
                    <leftOperandType>ConversationVariable</leftOperandType>
                    <operatorType>Equals</operatorType>
                    <rightOperandValue>false</rightOperandValue>
                </botStepConditions>
                <botSteps>
                    <botNavigation>
                        <botNavigationLinks>
                            <targetBotDialog>Customer_Verification</targetBotDialog>
                        </botNavigationLinks>
                        <type>Call</type>
                    </botNavigation>
                    <type>Navigation</type>
                </botSteps>
                <type>Group</type>
            </botSteps>
            <botSteps>
                <botStepConditions>
                    <leftOperandName>Location</leftOperandName>
                    <leftOperandType>ConversationVariable</leftOperandType>
                    <operatorType>IsNotSet</operatorType>
                </botStepConditions>
                <botSteps>
                    <botNavigation>
                        <botNavigationLinks>
                            <targetBotDialog>Select_Location</targetBotDialog>
                        </botNavigationLinks>
                        <type>Call</type>
                    </botNavigation>
                    <type>Navigation</type>
                </botSteps>
```

```
               <type>Group</type>
           </botSteps>
           <botSteps>
               <botVariableOperation>
                   <botInvocation>
                       <invocationActionName>CreateOrderService</invocationActionName>
                       <invocationActionType>apex</invocationActionType>
                       <invocationMappings>
                           <parameterName>customer</parameterName>
                           <type>Input</type>
                           <variableName>Contact</variableName>
                           <variableType>ConversationVariable</variableType>
                       </invocationMappings>
                       <invocationMappings>
                           <parameterName>location</parameterName>
                           <type>Input</type>
                           <variableName>Location</variableName>
                           <variableType>ConversationVariable</variableType>
                       </invocationMappings>
                       <invocationMappings>
                           <parameterName>output</parameterName>
                           <type>Output</type>
                           <variableName>Pizza_Order</variableName>
                           <variableType>ConversationVariable</variableType>
                       </invocationMappings>
                   </botInvocation>
                   <type>Set</type>
               </botVariableOperation>
               <type>VariableOperation</type>
           </botSteps>
           <botSteps>
               <botMessages>
                <message>Perfect, let&apos;s work on your order from our {!Location.Name}
location</message>
               </botMessages>
               <type>Message</type>
           </botSteps>
           <botSteps>
               <botNavigation>
                   <botNavigationLinks>
                       <targetBotDialog>Add_Items_to_Order</targetBotDialog>
                   </botNavigationLinks>
                   <type>Redirect</type>
               </botNavigation>
               <type>Navigation</type>
           </botSteps>
           <developerName>New_Order</developerName>
           <label>New Order</label>
           <mlIntent>New_Order</mlIntent>
           <showInFooterMenu>false</showInFooterMenu>
       </botDialogs>
       <conversationVariables>
           <dataType>Object</dataType>
           <developerName>Contact</developerName>
```

```
            <label>Contact</label>
        </conversationVariables>
        <conversationVariables>
            <dataType>Text</dataType>
            <developerName>Delivery_Address</developerName>
            <label>Delivery Address</label>
        </conversationVariables>
        <conversationVariables>
            <dataType>Object</dataType>
            <developerName>Pizza_Order</developerName>
            <label>Pizza Order</label>
        </conversationVariables>
        <entryDialog>Welcome</entryDialog>
        <mainMenuDialog>Main_Menu</mainMenuDialog>
    </botVersions>
    <label>Astro&apos;s Pizza</label>
</Bot>
```

The following is an example package.xml that references the previous definition.

```
<?xml version="1.0" encoding="UTF-8"?>
<Package xmlns="http://soap.sforce.com/2006/04/metadata">
    <types>
        <members>Astros Pizza_Bot.v1</members>
        <name>BotVersion</name>
    </types>
    <version>45.0</version>
</Package>
```

Wildcard Support in the Manifest File

This metadata type doesn't support the wildcard character * (asterisk) in the package.xml manifest file. For information about using the manifest file, see Deploying and Retrieving Metadata with the Zip File.

BrandingSet

Represents the definition of a set of branding properties for a Lightning community, as defined in the Theme panel in Experience Builder. This type extends the Metadata metadata type and inherits its fullName field.

File Suffix and Directory Location

BrandingSet components have the suffix brandingSet and are stored in the brandingSets folder.

Version

BrandingSet components are available in API version 40.0 and later.

Special Access Rules

The BrandingSet object is available when at least one of the following is enabled in your org: Salesforce Communities, Surveys, or Lightning Experience.

Fields

Field Name	Field Type	Description
brandingSetProperty	BrandingSetProperty[]	An array containing the name and value of each branding property, such as TextColor:#333.
description	string	A description of the set of branding properties.
masterLabel	string	Required. The user interface name of the set of branding properties.
type	string	The assigned branding set definition for this BrandingSet.

BrandingSetProperty

Represents the definition of a branding property in the Theme panel in Experience Builder.

Field Name	Field Type	Description
propertyName	string	Required. The name of the branding property, such as TextColor.
propertyValue	string	The value of the branding property, such as #333.

Declarative Metadata Sample Definition

The following is an example of a BrandingSet component.

```
<?xml version="1.0" encoding="UTF-8"?>
<BrandingSet xmlns="http://soap.sforce.com/2006/04/metadata">
    <brandingSetProperty>
        <propertyName>TextTransformStyle</propertyName>
        <propertyValue>uppercase</propertyValue>
    </brandingSetProperty>
    <brandingSetProperty>
        <propertyName>DetailTextColor</propertyName>
        <propertyValue>#696969</propertyValue>
    </brandingSetProperty>
    <brandingSetProperty>
        <propertyName>BorderColor</propertyName>
        <propertyValue>#D4D4D4</propertyValue>
    </brandingSetProperty>
    <brandingSetProperty>
        <propertyName>HeaderImage</propertyName>
        <propertyValue></propertyValue>
    </brandingSetProperty>
    <brandingSetProperty>
```

```
        <propertyName>HeaderFonts</propertyName>
        <propertyValue>Montserrat</propertyValue>
    </brandingSetProperty>
    <brandingSetProperty>
        <propertyName>CardBackgroundColor</propertyName>
        <propertyValue>rgba(255, 255, 255, 0)</propertyValue>
    </brandingSetProperty>
    <brandingSetProperty>
        <propertyName>LoginBackgroundColor</propertyName>
        <propertyValue>#F4F4F4</propertyValue>
    </brandingSetProperty>
    <brandingSetProperty>
        <propertyName>ActionColor</propertyName>
        <propertyValue>#2574A9</propertyValue>
    </brandingSetProperty>
    <brandingSetProperty>
        <propertyName>_ActionColorTrans</propertyName>
        <propertyValue>rgba(25, 124, 190, 0.9)</propertyValue>
    </brandingSetProperty>
    <brandingSetProperty>
        <propertyName>CompanyLogo</propertyName>
        <propertyValue></propertyValue>
    </brandingSetProperty>
    <brandingSetProperty>
        <propertyName>LoginBackgroundImage</propertyName>
        <propertyValue>../../../../sfsites/picasso/core/external/
            salesforceIdentity/images/background.jpg?v=1</propertyValue>
    </brandingSetProperty>
    <brandingSetProperty>
        <propertyName>_LinkColorDarker</propertyName>
        <propertyValue>#135F90</propertyValue>
    </brandingSetProperty>
    <brandingSetProperty>
        <propertyName>_ActionColorDarker</propertyName>
        <propertyValue>#135F90</propertyValue>
    </brandingSetProperty>
    <brandingSetProperty>
        <propertyName>_HoverColor</propertyName>
        <propertyValue>rgba(25, 124, 190, 0.1)</propertyValue>
    </brandingSetProperty>
    <brandingSetProperty>
        <propertyName>ErrorFontColor</propertyName>
        <propertyValue>#ff9e9e</propertyValue>
    </brandingSetProperty>
    <brandingSetProperty>
        <propertyName>TextColor</propertyName>
        <propertyValue>#333</propertyValue>
    </brandingSetProperty>
    <brandingSetProperty>
        <propertyName>OverlayTextColor</propertyName>
        <propertyValue>#FFFFFF</propertyValue>
    </brandingSetProperty>
    <brandingSetProperty>
        <propertyName>PrimaryFont</propertyName>
        <propertyValue>Lato</propertyValue>
    </brandingSetProperty>
    <brandingSetProperty>
        <propertyName>LinkColor</propertyName>
        <propertyValue>#2574A9</propertyValue>
    </brandingSetProperty>
    <masterLabel>ex</masterLabel>
    <type>napili:branding-napili-merged</type>
</BrandingSet>
```

The following is an example `package.xml` that references the previous definition.

```
<?xml version="1.0" encoding="UTF-8"?>
<Package xmlns="http://soap.sforce.com/2006/04/metadata">
    <types>
        <members>MyBrandingSet</members>
        <name>BrandingSet</name>
    </types>
    <version>40.0</version>
</Package>
```

Wildcard Support in the Manifest File

This metadata type supports the wildcard character `*` (asterisk) in the `package.xml` manifest file. For information about using the manifest file, see Deploying and Retrieving Metadata with the Zip File.

CallCenter

Represents the Call Center definition used to integrate Salesforce with a third-party computer-telephony integration (CTI) system.

File Suffix and Directory Location

CallCenter components have the suffix `callCenter` and are stored in the `callCenters` folder.

Version

CallCenter components are available in API version 27.0 and later.

Fields

Field Name	Field Type	Description
`adapterUrl`	string	Optional field. A URL that points to an adapter.
`displayName`	string	The display name of this call center.
`displayNameLabel`	string	The label of the `displayName` field in Call Center setup page.
`internalNameLabel`	string	The label of the `internalName` field in Call Center setup page.

Field Name	Field Type	Description
version	string	The version of this call center.
sections	CallCenterSection[]	Custom setup items defined for this call center.

CallCenterSection

Field Name	Field Type	Description
items	CallCenterItem[] on page 230	Contains the label, name, and value that describe the sections.
label	string	The label of the section.
name	string	The name of the section.

CallCenterItem

Field Name	Field Type	Description
label	string	The label of the custom setup item.
name	string	The name of the custom setup item.
value	int or URL	The value of the custom setup item.

Declarative Metadata Sample Definition

The following is an example of a CallCenter component:

```
<?xml version="1.0" encoding="UTF-8"?>
<CallCenter xmlns="http://soap.sforce.com/2006/04/metadata">
    <adapterUrl>http://localhost:11000</adapterUrl>
    <displayName>Demo Call Center Adapter</displayName>
    <displayNameLabel>Display Name</displayNameLabel>
    <internalNameLabel>Internal Name</internalNameLabel>
    <sections>
        <items>
            <label>Description</label>
            <name>reqDescription</name>
            <value>Demo Call Center Adapter</value>
        </items>
        <items>
            <label>CTI Connector ProgId</label>
            <name>reqProgId</name>
            <value>DemoAdapter.DemoAdapter.1</value>
        </items>
        <items>
            <label>Version</label>
            <name>reqVersion</name>
            <value>3.0</value>
        </items>
        <items>
            <label>CTI Adapter URL</label>
            <name>reqAdapterUrl</name>
            <value>http://localhost:11000</value>
        </items>
        <label>General Information</label>
        <name>reqGeneralInfo</name>
    </sections>
    <sections>
        <items>
            <label>Outside Prefix</label>
            <name>reqOutsidePrefix</name>
            <value>1</value>
        </items>
        <items>
            <label>Long Distance Prefix</label>
            <name>reqLongDistPrefix</name>
            <value>1</value>
        </items>
        <items>
            <label>International Prefix</label>
            <name>reqInternationalPrefix</name>
            <value>01</value>
        </items>
        <label>Dialing Options</label>
        <name>reqDialingOptions</name>
    </sections>
    <version>4</version>
</CallCenter>
```

Wildcard Support in the Manifest File

This metadata type supports the wildcard character `*` (asterisk) in the `package.xml` manifest file. For information about using the manifest file, see Deploying and Retrieving Metadata with the Zip File.

CampaignInfluenceModel

Represents a campaign influence model used by Customizable Campaign Influence. You can’t configure Customizable Campaign Influence via the Metadata API, but you can add a campaign influence model.

Note: This information applies only to Customizable Campaign Influence and not to Campaign Influence 1.0.

This type extends the Metadata metadata type and inherits its `fullName` field.

File Suffix and Directory Location

CampaignInfluenceModel values are stored in the `campaignInfluenceModels` directory of the corresponding package directory. The file name matches the model name, and the extension is `.campaignInfluenceModel`.

Version

CampaignInfluenceModel components are available in API version 38.0 and later.

Fields

Field Name	Field Type	Description
`isActive`	boolean	Indicates whether the model is active. Active models can generate campaign influence records. Deactivating a model deletes its campaign influence records. Custom models are always active and this field is ignored. This field is available beginning with API version 40.0.
`isDefaultModel`	boolean	Required. Indicates if the model is the default model or not. Only campaign influence records associated with the default model appear on campaigns and opportunities. You can only have one default model at a time. A model must be active to become the default model. Activating or deactivating custom models does not automatically generate or delete campaign influence records.
`isModelLocked`	boolean	Required. Indicates if the model is locked or not. Campaign Influence records for locked models can be manipulated only via the API.
`modelDescription`	string	A description of the influence model.
`name`	string	Required. A unique name for the model.
`recordPreference`	picklist	The value of this field determines when to create campaign influence records. • `AllRecords`: Creates records regardless of the revenue attribution percentage. • `RecordsWithAttribution`: Creates records only when the revenue attribution is greater than 0%. This field is available In API version 41.0 and later.

Declarative Metadata Sample Definition

The following is an example of a CampaignInfluenceModel component that represents the default Salesforce campaign influence attribution model. The default `isDefaultModel` value of `true` can be changed if another model is created and set as the default model. The `isModelLocked` value of `true` means that Campaign Influence records for this model can be seen in the UI, but not created, updated, or deleted.

```
<?xml version="1.0" encoding="UTF-8"?>
<CampaignInfluenceModel xmlns="http://soap.sforce.com/2006/04/metadata">
    <isActive>true</isActive>
    <isDefaultModel>true</isDefaultModel>
   <isModelLocked>true</isModelLocked>   <recordPreference>AllRecords</recordPreference>
```

```
    <modelDescription>Primary Campaign gets 100% of the revenue share</modelDescription>
    <name>Salesforce Model</name>
</CampaignInfluenceModel>
```

The following is an example of a CampaignInfluenceModel component that creates an influence model called Last Touch, which will not be the default model.

```
<?xml version="1.0" encoding="UTF-8"?>
<CampaignInfluenceModel xmlns="http://soap.sforce.com/2006/04/metadata">
    <isActive>true</isActive>
    <isDefaultModel>false</isDefaultModel>
    <isModelLocked>true</isModelLocked>
    <modelDescription>This model gives 100% influence attribution to the last campaign
that touched the contact.</modelDescription>
    <name>Last Touch</name>
    <recordPreference>RecordsWithAttribution</recordPreference>
</CampaignInfluenceModel>
```

Wildcard Support in the Manifest File

This metadata type supports the wildcard character `*` (asterisk) in the `package.xml` manifest file. For information about using the manifest file, see Deploying and Retrieving Metadata with the Zip File.

CaseSubjectParticle

Represents the Social Business Rules custom format for the **Case Subject** field on cases created from inbound social posts.

File Suffix and Directory Location

CaseSubjectParticle components have the suffix `.CaseSubjectParticle` and are stored in the `CaseSubjectParticles` folder.

Version

CaseSubjectParticle is available in API version 41.0 and later.

Fields

Field Name	Field Type	Description
`index`	int	Required. The order in which the custom **Case Subject** is generated, meaning if the social network is 0 and the social message is 1, then the subject generates as `Twitter	Tweet`.
`textField`	string	Specifies inbound social content added to **Case Subject** in case records.

Field Name	Field Type	Description
`type`	CaseSubjectParticleType (enumeration of type string)	Required. Specifies the custom **Case Subject** format from which inbound social content appears in case records. Valid values are: • `ProvidedString` • `Source` • `MessageType` • `SocialHandle` • `SocialNetwork` • `Sentiment` • `RealName` • `Content` • `PipeSeparator` • `ColonSeparator` • `HyphenSeparator`

Declarative Metadata Sample Definition

This is a sample of a `.CaseSubjectParticle` file.

```
<?xml version="1.0" encoding="UTF-8"?>
<Package xmlns=http://soap.sforce.com/2006/04/metadata"">
    <types>
        <members>*</members>
        <name>CaseSubjectParticle</name>
    </types>
    <version>41.0</version>
</Package>
```

Wildcard Support in the Manifest File

This metadata type doesn't support the wildcard character `*` (asterisk) in the `package.xml` manifest file. For information about using the manifest file, see Deploying and Retrieving Metadata with the Zip File.

Certificate

Represents a certificate used for digital signatures which verify that requests are coming from your org. Certificates are used for either authenticated single sign-on with an external website, or when using your org as an identity provider. This type extends the MetadataWithContent metadata type and inherits its `content` and `fullName` fields.

File Suffix and Directory Location

Certificate components have the suffix `.crt` and are stored in the `certs` folder.

Version

Certificate components are available in API version 36.0 and later.

Fields

Field Name	Field Type	Description
`caSigned`	boolean	Required. Indicates whether this certificate is signed by the issuer (true) or not (false).
`encryptedWithPlatformEncryption`	boolean	Indicates whether this certificate is encrypted with Platform Encryption.
`expirationDate`	dateTime	The date that this certificate expires and is no longer usable. For self-signed certificates, if `keySize` is 2048 bits, the expiration date is automatically 1 year after you create the certificate. If `keySize` is 4096 bits, the expiration date is automatically 2 years after you create the certificate. For CA-signed certificates, `expirationDate` is automatically updated to the signed certificate's expiration date when a signed certificate chain is uploaded. The date format is YYYY-MM-DD.
`keySize`	int	Certificate keys can be either 2048 bits or 4096 bits. A certificate with 4096-bit keys lasts 2 years, and a certificate with 2048-bit keys lasts 1 year. Certificates with 2048-bit keys are faster than certificates with 4096-bit keys. If `keySize` isn't specified when you create a certificate, the key size defaults to 2048 bits.
`masterLabel`	string	Required. A user-friendly name for the certificate that appears in the Salesforce user interface, such as in Certificate and Key Management. Limit: 64 characters.
`privateKeyExportable`	boolean	Indicates whether this certificate's private key is exportable. If `privateKeyExportable` isn't specified when you create a certificate, its default value is `true`.

Usage

The Metadata API can be used to create a self-signed or a CA-signed certificate. The .crt file's contents are the certificate chain, which can be updated when you renew or update the intermediate certificate chain of a CA-signed certificate. After creating a CA-signed certificate, the .crt file contains a certificate signing request (CSR). For details, see About Salesforce Certificates and Keys in the Salesforce Help.

To copy an existing certificate's X.509 parameter data to a new certificate, upload the existing certificate. You can also use this procedure to renew a certificate. A new private+public key pair is created with a new certificate. Salesforce doesn't allow the import or export of the private key via the API.

Using the Metadata API, you can download a CSR. After it's CA-signed, you can upload it back to Salesforce.

- **Downloading a CSR.** The CSR is downloadable after a CA-signed cert is created. If a signed certificate hasn't been uploaded to that certificate, the content of the downloaded .crt file is the CSR.
- **Uploading a CA-Signed Certificate.** To upload the signed certificate chain back to Salesforce, save the signed certificate chain as the content of the .crt file and update it via the Metadata API.

Note: After the signed certificate chain is uploaded via the Metadata API, the CSR of that certificate can't be downloaded via the API anymore. This is because the content of the .crt file is the signed certificate chain. However, the CSR can still be downloaded via the UI.

Declarative Metadata Sample Definition

The following is an example of a Certificate component.

```
<?xml version="1.0" encoding="UTF-8"?>
<Certificate xmlns="http://soap.sforce.com/2006/04/metadata">
    <caSigned>true</caSigned>
    <encryptedWithPlatformEncryption>true</encryptedWithPlatformEncryption>
    <expirationDate>2017-03-19</expirationDate>
    <keySize>4096</keySize>
    <masterLabel>My Certificate Name</masterLabel>
    <privateKeyExportable>true</privateKeyExportable>
</Certificate>
```

Wildcard Support in the Manifest File

This metadata type supports the wildcard character `*` (asterisk) in the `package.xml` manifest file. For information about using the manifest file, see Deploying and Retrieving Metadata with the Zip File.

ChatterExtension

Represents the metadata used to describe a Rich Publisher App that's integrated with the Chatter publisher.

Retrieving ChatterExtension

Using Workbench or another API tool, you can get extension information from `package.xml` using this code.

```
<?xml version="1.0" encoding="UTF-8"?>
<Package xmlns="http://soap.sforce.com/2006/04/metadata">
 <types>
  <members>xw1</members>
  <name>ChatterExtension</name>
 </types>
 <version>41.0</version>
</Package>
```

Use the `<members>` tag to name a specific extension (in this example, *`xw1`*), or use the wildcard (*) symbol to retrieve all your extensions.

Here's an example of retrieved information.

```
<?xml version="1.0" encoding="UTF-8"?>
<ChatterExtension xmlns="http://soap.sforce.com/2006/04/metadata">
  <compositionComponent>xwComp</compositionComponent>
  <description>des</description>
  <extensionName>xw1</extensionName>
  <headerText>h1</headerText>
  <hoverText>h2</hoverText>
  <icon>tiger</icon>
  <masterLabel>master</masterLabel>
  <renderComponent>xwRend</renderComponent>
  <type>Lightning</type>
</ChatterExtension>
```

Version

ChatterExtension is a new feature in API version 41.0.

Fields

Field	Field Type	Description
`compositionComponent`	string	Required. The composition component of the Rich Publisher App that you provide. It's comprised of the `lightning:availableForChatterExtensionComposer` interface.
`description`	string	Required. The description of your custom Rich Publisher App.
`extensionName`	string	Required. The name of your extension. That is, your Rich Publisher App.
`headerText`	string	The text to show in the header of your app composer. Header text is required for Lightning type extensions. This text can be localized.
`hoverText`	string	The text to show when a user mouses over your extension's icon. Mouse-over text is required for Lightning type extensions. This text can be localized.
`icon`	string	Required. The icon to show in the Chatter publisher. Use an existing file asset id from your org.
`isProtected`	boolean	An auto-generated value. It currently has no impact.
`masterLabel`	string	Required. Master label for the ChatterExtension object.
`renderComponent`	string	Required. The rendering component of the Rich Publisher App that you provide. It's comprised of the `lightning:availableForChatterExtensionRenderer` interface.
`type`	ChatterExtensionType (enumeration of type string)	Required. Describes the type of the extension. Currently, the only value supported is *`Lightning`*. Included to allow for other possible types in the future.

Wildcard Support in the Manifest File

This metadata type supports the wildcard character `*` (asterisk) in the `package.xml` manifest file. For information about using the manifest file, see Deploying and Retrieving Metadata with the Zip File.

SEE ALSO:

Integrate Your Custom Apps into the Chatter Publisher

CleanDataService

Represents a data service that adds and updates data in standard objects. This type extends the Metadata metadata type and inherits its `fullName` field.

File Suffix and Directory Location

CleanDataService components have the `.cleanDataService` suffix and are stored in the `cleanDataServices` directory. The name of the component file is based on the name of the object associated with the data service. For example, the component file name `cleanDataServices/DataCloudCompanyMatch.cleanDataService` describes a data service component called DataCloudCompanyMatch that's associated with the company name in account objects.

Version

CleanDataService components are available in API version 48.0 and later.

Fields

Field Name	Field Type	Description
`cleanRules`	CleanRule[]	Required. A list of data integration rules
`description`	string	Required. A description of the data service
`masterLabel`	string	Required. Master label for this data service. Although this value is displayed, it’s an internal label for the data service and isn’t translated.
`matchEngine`	string	Required. A key that maps to the internal data service identifier.

CleanRule

Represents information that controls how the data service adds and updates data in an org.

Field Name	Field Type	Description
`bulkEnabled`	boolean	Required. If this field is set to `true`, Salesforce applies the data integration rule to existing records whenever the rule is updated or saved.
`bypassTriggers`	boolean	Required. If this field is set to `true`, Salesforce bypasses triggers when it applies the rule; otherwise, it applies triggers after it applies the rule.

Field Name	Field Type	Description
`bypassWorkflow`	boolean	Required. If this field is set to `true`, Salesforce bypasses workflow rules when it applies the data integration rule; otherwise, it applies workflow rules after it applies the rule.
`description`	string	Required. User-friendly text that describes the data integration rule.
`developerName`	string	Required. This name can contain only underscores and alphanumeric characters, and must be unique in your org. It must begin with a letter, not include spaces, not end with an underscore, and not contain two consecutive underscores. This unique name prevents conflicts with rules from other packages that have the same MasterLabel.
`fieldMappings`	FieldMapping[]	Required. A list of FieldMapping entries for the rule.
`masterLabel`	string	Required. Master label for this object. This display value is the internal label that is not translated.
`matchRule`	string	Required. An internal label for a matching rule in the data service that's associated with the CleanRule.
`sourceSobjectType`	string	Required. A virtual object in the data service that is associated with the CleanRule. Specifying a non-existent object causes an error.
`status`	string	Required. Status of the data integration rule. Valid values are `Active` and `Inactive`.
`targetSobjectType`	string	Required. A standard object that's the target of additions and updates specified by this CleanRule. Specifying an object that the data service does not support causes an error.

FieldMapping

Represents a mapping between fields in the data service and fields in an object in the org.

Field Name	Field Type	Description
`developerName`	string	Required. This name can contain only underscores and alphanumeric characters, and must be unique in your org. It must begin with a letter, not include spaces, not end with an underscore, and not contain two consecutive underscores. This unique name prevents conflicts with field mappings from other packages that have the same MasterLabel.
`fieldMappingRows`	FieldMappingRow[]	Required. A list of FieldMappingRow entries. Each entry represents a field in a standard object that maps to a field in the data service.
`masterLabel`	string	Required. Master label for this object. This display value is the internal label that is not translated.
`SObjectType`	string	Required. The standard object associated with this FieldMapping. Specifying an object that the data service does not support causes an error.

FieldMappingRow

Represents the status of a CleanRule.

Field Name	Field Type	Description
`fieldName`	string	The display name for the field represented by the FieldMappingRow.
`fieldMappingFields`	FieldMappingField[]	Required. A list of FieldMappingField entries. Each entry is a field in a standard object that maps to a field in the data service.
`mappingOperation`	string	The comparison operation the data service applies when it compares the value of this FieldMappingRow to the mapped field in the object specified in SObjectType. The value of this field is `AutoFill`, which indicates that the data service only adds data if the object field is blank.
`SObjectType`	string	The standard object for the field mapped to the FieldMappingRow. Specifying an object that the data service does not support causes an error.

FieldMappingField

Represents a field in a standard object. A FieldMappingField maps to a FieldMappingRow entry in a data service.

Field Name	Field Type	Description
`dataServiceField`	string	Required. A field in the data service that is mapped to this field.
`dataServiceObjectName`	string	Required. An object in the data service that contains the FieldMappingRow associated with this FieldMappingField. Specifying a non-existent object causes an error.
`priority`	int	Required. Represents the priority that the data service uses when it updates the field, relative to other update rules for the same field. Valid values are 1-100.

Declarative Metadata Sample Definition

The following is an example of a CleanDataService component for the lead standard object.

```xml
<?xml version="1.0" encoding="UTF-8"?>
<CleanDataService xmlns="http://soap.sforce.com/2006/04/metadata">
    <cleanRules>
        <bulkEnabled>false</bulkEnabled>
        <bypassTriggers>false</bypassTriggers>
        <bypassWorkflow>false</bypassWorkflow>
        <description>Adds data info to leads</description>
        <developerName>DataService_Leads_Enrichment</developerName>
        <fieldMappings>
            <SObjectType>DataServiceCompanyObject</SObjectType>
            <developerName>DataService_Leads_Enrichment_InputMapping</developerName>
            <fieldMappingRows>
                <SObjectType>DataServiceCompanyObject</SObjectType>
                <fieldMappingFields>
```

```
                    <dataServiceField>Email</dataServiceField>
                    <dataServiceObjectName>Lead</dataServiceObjectName>
                    <priority>1</priority>
                </fieldMappingFields>
                <fieldName>Email</fieldName>
                <mappingOperation>Autofill</mappingOperation>
            </fieldMappingRows>
            <fieldMappingRows>
                <SObjectType>DataServiceCompanyObject</SObjectType>
                <fieldMappingFields>
                    <dataServiceField>Company</dataServiceField>
                    <dataServiceObjectName>Lead</dataServiceObjectName>
                    <priority>1</priority>
                </fieldMappingFields>
                <fieldName>Name</fieldName>
                <mappingOperation>Autofill</mappingOperation>
            </fieldMappingRows>
            <masterLabel>DataServiceInputMapping</masterLabel>
        </fieldMappings>
        <fieldMappings>
            <SObjectType>Lead</SObjectType>
            <developerName>DataService_Leads_Enrichment_OutputMapping</developerName>
            <fieldMappingRows>
                <SObjectType>Lead</SObjectType>
                <fieldMappingFields>
                    <dataServiceField>EmployeesTotal</dataServiceField>
                  <dataServiceObjectName>DataServiceCompanyObject</dataServiceObjectName>

                    <priority>1</priority>
                </fieldMappingFields>
                <fieldName>NumberOfEmployees</fieldName>
                <mappingOperation>Autofill</mappingOperation>
            </fieldMappingRows>
            <fieldMappingRows>
                <SObjectType>Lead</SObjectType>
                <fieldMappingFields>
                    <dataServiceField>Revenue</dataServiceField>
                  <dataServiceObjectName>DataServiceCompanyObject</dataServiceObjectName>

                    <priority>1</priority>
                </fieldMappingFields>
                <fieldName>AnnualRevenue</fieldName>
                <mappingOperation>Autofill</mappingOperation>
            </fieldMappingRows>
            <fieldMappingRows>
                <SObjectType>Lead</SObjectType>
                <fieldMappingFields>
                    <dataServiceField>Industry</dataServiceField>
                  <dataServiceObjectName>DataServiceCompanyObject</dataServiceObjectName>

                    <priority>1</priority>
                </fieldMappingFields>
                <fieldName>Industry</fieldName>
                <mappingOperation>Autofill</mappingOperation>
            </fieldMappingRows>
            <masterLabel>DataServiceOutputMapping</masterLabel>
        </fieldMappings>
        <masterLabel>Data Service Company Info for Leads</masterLabel>
        <matchRule>DataServiceLeadAppendMatchRule</matchRule>
        <sourceSobjectType>DataServiceCompanyObject</sourceSobjectType>
        <status>Active</status>
        <targetSobjectType>Lead</targetSobjectType>
    </cleanRules>
    <description>Data Service Companies for Leads</description>
    <masterLabel>Data Service Companies for Leads</masterLabel>
    <matchEngine>LeadEnrichmentMatchEngine</matchEngine>
</CleanDataService>
```

The following is an example `package.xml` that references the previous definition.

```
<?xml version="1.0" encoding="UTF-8"?>
<Package xmlns="http://soap.sforce.com/2006/04/metadata">
    <types>
        <members>DataService_Leads_Enrichment</members>
        <name>CleanDataService</name>
    </types>
    <version>38.0</version>
</Package>
```

Usage

Use CleanDataService to retrieve all the metadata that describes a data enrichment service. To configure the service in a new org, deploy the metadata you retrieved. Avoid using CRUD-Based Calls with CleanDataService.

To make small modifications to the CleanDataService component, use the Tooling API.

Wildcard Support in the Manifest File

This metadata type supports the wildcard character `*` (asterisk) in the `package.xml` manifest file. For information about using the manifest file, see Deploying and Retrieving Metadata with the Zip File.

CMSConnectSource

Represents the connection information for external content management systems that feed content to Communities. This type extends the Metadata metadata type and inherits its `fullName` field.

Note: For use with Change Sets, CMSConnectSource is a dependent of **Network** and **Community**.

File Suffix and Directory Location

CMSConnectSource components have the suffix `.cmsConnectSource` and are stored in the `cmsConnectSource` folder. In that folder, separate files exist for each network (for example, `*networkname*.*sourcedevelopername*.cmsConnectSource`). Each file represents a CMS connection.

Version

CMSConnectSource components are available in API version 43.0 and later.

Special Access Rules

The **CMS Connect org** permission must be enabled.

Fields

Field Name	Field Type	Description
`cmsConnectAsset`	CMSConnectAsset on page 244[]	Represents CSS or JavaScript defined for the connection. • 0–10 for CSS • 0–10 for JavaScript
`cmsConnectLanguage`	CMSConnectLanguage on page 245[]	0 to more. Represents language mappings defined for the connection.
`cmsConnectPersonalization`	CMSConnectPersonalization[] on page 245	0 or 1. This represents personalization defined for the connection. Only for use when `type` is `AEM`.
`cmsConnectResourceType`	CMSConnectResourceType on page 245[]	0–5. This represents JSON definitions defined for the connection.
`connectionType`	CMSSourceConnectionType (enumeration of type string)	Required. Type of authentication being used with outside system. Valid values are: • `Public` • `Authenticated`
`cssScope`	string	The class name used to prefix and scope the CSS rules.
`developerName`	string	Required. API name of the CMSConnectSource entity.
`languageEnabled`	string	Required. Valid values are: • `Y` to enable language mapping for connection. • `N` if no language mapping is needed.
`masterLabel`	string	Required. Connection name
`namedCredential`	string	Required when the `connectionType` is `Authenticated`. API name of `namedCredential`. Before deploying `namedCredential`, it must exist on the destination org.
`personalizationEnabled`	string	Required. Valid values are: • `Y` to enable personalization mapping for connection. • Otherwise `N`.
`rootPath`	string	Required. Root path.

Field Name	Field Type	Description
`sortOrder`	int	Required. Defines the load order of the connection when multiple connections defined on page. The load order begins with 1.
`status`	CMSConnectionStatus (enumeration of type string)	Required. Status of connection. Valid values are: • `ACTIVE` • `INACTIVE`
`type`	CMSConnectionSourceType (enumeration of type string)	Required. The identification of the source connection system. Valid values are: • `AEM` • `Drupal` • `WordPress` • `SDL` • `Sitecore` • `Other`
`websiteUrl`	string	Required if `connectionType` is `Public`

Note: Because there may be existing connections when a package comes in, there is some INSERT or UPDATE logic that should be taken into account:

- If `developerName` is found in the destination, then update the existing collection with all details form source.
- `namedCredential` is handled through `developerName`. If `namedCredential` with `developerName` is not found, then an error is generated.
- If `sortOrder` from the source is not in the destination, then insert/update with the source `sortOrder`.
- If `sortOrder` from the source is already in the destination, then increase the source `sortOrder` by 1 for connections such that the destination `sortOrder` > `sortOrder` from the source.

CMSConnectAsset

CMSConnectAsset defines the location, types, and order of assets necessary to support the incoming content, such as JavaScript and CSS files.

Note: Because there may be existing connections when a package comes in, there is some INSERT or UPDATE logic that should be taken into account:

- If `assetPath` exists in the destination, then update the existing record, else the new `assetPath` will be inserted.
- Always keep the `sortOrder` from the source and adjust the destination accordingly.

Field Name	Field Type	Description
`assetPath`	string	Relative path of the asset.
`assetType`	string	When used in Apex, this can be sent as an enum, otherwise, this has a field type of string.

Field Name	Field Type	Description
		Allowed values as string - `CSS` - `Javascript` Allowed values as enum - `CSS` - `Javascript`
`sortOrder`	int	Loading sequence on the page.

CMSConnectLanguage

CMSConnectLanguage components determine the presented language of the content.

Field Name	Field Type	Description
`cmsLanguage`	string	When a language placeholder is in the URL path, this value is used to replace it.
`language`	string	Salesforce supported language. Note: For information see https://developer.salesforce.com/docs/atlas.en-us.api_meta.meta/api_meta/meta_translations.htm

CMSConnectPersonalization

CMSConnectPersonalization is only used in conjunction with Adobe Experience Manager (AEM).

Note: Because there may be existing connections when a package comes in, there is some INSERT or UPDATE logic that should be taken into account. If personalization is not enabled in the source system, but it is enabled in the destination, the destination is disabled. The record for the connection is deleted from the table.

Field Name	Field Type	Description
`connectorPage`	string	The path to the JSP file that you created and installed in AEM.
`connectorPageAsset`	string	The path to your Javascript file. Providing this path allows you to run scripts dynamically.

CMSConnectResourceType

CMSConnectResourceType is for use only to define JSON connections.

Note: Because there may be existing connections when a package comes in, there is some INSERT or UPDATE logic that should be taken into account. If the developer name was found in the destination, then update the existing record with all details from the source.

Field Name	Field Type	Description
cmsConnectResourceDefinition	cmsConnectResourceDefinition on page 246[]	0–10 allowed per CMSConnectResourceType.
developerName	string	API name of CMSConnectResourceType.
masterLabel	string	Content type name.
resourceType	string	The only allowed value is JSON.

CMSConnectResourceDefinition

cmsConnectResourceDefinition is used to define JSON connections.

Note: Because there may be existing connections when a package comes in, there is some INSERT or UPDATE logic that should be taken into account:

- If developerName is found in the destination, then the existing record is updated with all details from the new source, else the new value is inserted.
- If the current source is DETAIL and the destination has DETAIL with a different name, then the destination is updated to LIST and the source inserted as DETAIL.

Field Name	Field Type	Description
developerName	string	Required. API name of CMSConnectResourceDefinition.
masterLabel	string	Required. developerName of Content Item or Content List.
options	int	Required. Identifies whether the content from the external source is a single item or a list. 0 for Content List 1 for Content Item
payloadType	string	Required. The only valid value is JSON.
resourceIdPath	string	Relative path to ID. Required for Content Item.
resourceNamePath	string	Relative path to resource name. Required for Content Item.
resourcePath	string	Required. JSON resource path.
rootNodePath	string	Only for Content List and collection. Defines the initial starting path for a collection or list.

Declarative Metadata Sample Definition

The following is an example of a CMSConnectSource definition.

```
<?xml version="1.0" encoding="UTF-8"?>
<CMSConnectSource xmlns="http://soap.sforce.com/2006/04/metadata">
    <cmsConnectAsset>
```

```
        <assetPath>etc/designs/capricorn/app-prefixed.min.css</assetPath>
        <assetType>CSS</assetType>
        <sortOrder>1</sortOrder>
    </cmsConnectAsset>
    <cmsConnectAsset>
        <assetPath>etc/designs/capricorn/w3data.js</assetPath>
        <assetType>Javascript</assetType>
        <sortOrder>1</sortOrder>
    </cmsConnectAsset>
    <cmsConnectLanguage>
        <cmsLanguage>en</cmsLanguage>
        <language>en_US</language>
    </cmsConnectLanguage>
    <cmsConnectLanguage>
        <cmsLanguage>fr</cmsLanguage>
        <language>fr</language>
    </cmsConnectLanguage>
    <cmsConnectPersonalization>
        <connectorPage>content/salesforceConnector.js</connectorPage>
        <connectorPageAsset>content/js/capricorn/assets.js</connectorPageAsset>
    </cmsConnectPersonalization>
    <cmsConnectResourceType>
        <cmsConnectResourceDefinition>
            <developerName>Details</developerName>
            <masterLabel>Details</masterLabel>
            <options>0</options>
            <payloadType>JSON</payloadType>
            <resourceIdPath>ID</resourceIdPath>
            <resourceNamePath>title</resourceNamePath>

<resourcePath>rest/v1.1/sites/cmstry.wordpress.com/posts/{component}</resourcePath>
        </cmsConnectResourceDefinition>
        <cmsConnectResourceDefinition>
            <developerName>List</developerName>
            <masterLabel>List</masterLabel>
            <options>1</options>
            <payloadType>JSON</payloadType>

<resourcePath>rest/v1.1/sites/cmstry.blog.wordpress.com/posts?number={itemsPerPage}&amp;page={pageNumber}</resourcePath>

        </cmsConnectResourceDefinition>
        <developerName>Posts</developerName>
        <masterLabel>Posts</masterLabel>
        <resourceType>JSON</resourceType>
    </cmsConnectResourceType>
    <connectionType>Public</connectionType>
    <cssScope>capricorn</cssScope>
    <developerName>Capricorn</developerName>
    <languageEnabled>Y</languageEnabled>
    <masterLabel>Capricorn</masterLabel>
    <personalizationEnabled>Y</personalizationEnabled>
    <rootPath>content/capricorn/{language}</rootPath>
    <sortOrder>11</sortOrder>
    <status>ACTIVE</status>
    <type>AEM</type>
    <websiteUrl>https://public-api.wordpress.com</websiteUrl>
</CMSConnectSource>
```

The following is an example `package.xml`.

```
<Package xmlns="http://soap.sforce.com/2006/04/metadata">
    <types>
        <members>NetworkA.*</members>
        <name>CMSConnectSource</name>
    </types>
    <version>43.0</version>
</Package>
```

To retrieve a specific connection:

```

<Package xmlns="http://soap.sforce.com/2006/04/metadata">
    <types>
        <members>NetworkA.DeveloperName</members>
        <name>CMSConnectSource</name>
    </types>
    <version>43.0</version>
</Package>
```

Usage

The INSERT or UPDATE logic for the incoming information is always UPSERT. If data is not in the entity, then it is inserted, otherwise the current data is updated.

Before doing upsert, the content from the package is validated against the maximum limits for the following:

- CSS assets <= 10
- JavaScript assets <= 10
- Resource types < =5
- Resource definitions for each type <=10

For example

1. The validation on a new connection only totals the elements in the incoming package.
2. Validation of existing connections totals the existing assets and new elements to assess validity. For example, if a connection on the destination org already has six CSS definitions, and the incoming package has defined seven CSS definitions (four new + three existing). The new total will be the six current from the database, ignoring the three repeated in the package, and adds four new definitions from the incoming package, totaling 10 definitions, which is at or below the 10 asset threshold and it passes validation.

Refer below for more details for each entity how is handled while saving the details from package to destination org:

Type	Description
`CMSConnectSource`	• If `developerName` is found in the destination, then update the existing collection with all details form source. • `namedCredential` is handled through `developerName`. If `namedCredential` with `developerName` is not found, then an error is generated.

Type	Description
	- If sortOrder from the source is not in the destination, then insert/update with the source sortOrder. - If sortOrder from the source is already in the destination, then increase the source sortOrder by 1 for connections such that the destination sortOrder > sortOrder from the source.
CMSConnectAsset	- If assetPath exists in the destination, then update the existing record, else the new assetPath will be inserted. - Always keep the sortOrder from the source and adjust the destination accordingly.
CMSConnectPersonalization	If personalization is not enabled in the source system, but it is enabled in the destination, the destination is disabled. The record for the connection is deleted from the table.
CMSConnectResourceType	If the developer name was found in the destination, then update the existing record with all details from the source.
CMSConnectResourceDefinition	- If developerName is found in the destination, then the existing record is updated with all details from the new source, else the new value is inserted. - If the current source is DETAIL and the destination has DETAIL with a different name, then the destination is updated to LIST and the source inserted as DETAIL.

Wildcard Support in the Manifest File

This metadata type supports the wildcard character `*` (asterisk) in the `package.xml` manifest file. For information about using the manifest file, see Deploying and Retrieving Metadata with the Zip File.

SEE ALSO:

Select Components for an Outbound Change Set

View and Add Dependent Components to a Change Set

Developer Guide: Deploying and Retrieving Metadata

Salesforce Help: Use Personalized Content in CMS Connect

Developer Guide: Translations

Community (Zone)

Represents a zone that contains Ideas or Chatter Answers objects. Zones are shared by the Ideas, Answers, and Chatter Answers features, allowing you to view and create zones from those locations.This type extends the Metadata metadata type and inherits its `fullName` field.

Note: Starting with the Summer '13 release, Chatter Answers and Ideas "communities" have been renamed to "zones." In API version 28, the API object label has changed to `Zone`, but the API type is still `Community`.

File Suffix and Directory Location

Zones have the suffix `community` and are stored in the `communities` folder.

Version

Community (Zone) components are available in API version 27.0 and later.

Fields

Note: When `enableChatterAnswers` is set to `false`, values specified for the following fields are ignored and not saved: `communityFeedPage`, `emailFooterDocument`, `emailHeaderDocument`, `enablePrivateQuestions`, `emailNotificationUrl`, and `site`.

Field Name	Field Type	Description
`active`	boolean	Indicates whether the zone is active (`true`) or not (`false`).
`chatterAnswersFacebookSsoUrl`	string	(Read only) The Facebook sign-on URL, which is based on the Facebook authentication provider selected in your Chatter Answers settings. This field is available only if Chatter Answers and Facebook Single Sign-On for Chatter Answers are enabled.
`communityFeedPage`	string	The Visualforce page that hosts the zone's feeds. This field is available when Chatter Answers is enabled in the organization.
`description`	string	The description of the zone.
`emailFooterDocument`	string	The text or HTML file that incorporates your organization's branding into the footer of email notifications. This field is available when Chatter Answers is enabled in the organization.
`emailHeaderDocument`	string	The text or HTML file that incorporates your organization's branding into the header of email notifications. This field is available when Chatter Answers is enabled in the organization.
`emailNotificationUrl`	string	The URL that's included in email notifications. This field is available when Chatter Answers is enabled in the organization. This field replaces `portalEmailNotificationUrl` in API version 28.0 and later.
`enableChatterAnswers`	boolean	Indicates whether the zone has Chatter Answers enabled (`true`) or not (`false`). This field is available when Chatter Answers is enabled in the organization.
`enablePrivateQuestions`	boolean	Indicates whether Chatter Answers questions can be escalated to cases (`true`) or not (`false`). This field is available when Chatter Answers is enabled in the organization.

Field Name	Field Type	Description
expertsGroup	string	The name of the public group that act as experts in the zone. This field is available when eitherIdeas or Answers are enabled in the organization.
portal	string	The name of the portal in which to display the zone.
portalEmailNotificationUrl	string	The portal URL that's included in email notifications. This field is available when Chatter Answers is enabled in the organization. This field has been replaced by `emailNotificationUrl` in API version 28.0 and later.
reputationLevels	ReputationLevels	The fields that define the points and name of each reputation level you define. You can create up to 25 reputation levels per zone.
showInPortal	boolean	Indicates whether the zone is available to all portals (`true`) or not available to any portals (`false`).
site	string	The name of the site for the zone. This field is available when Chatter Answers is enabled in the organization.

ReputationLevels

Represents the points and reputation label that displays on hover over a user's photo in the feed.

Field Name	Field Type	Description
chatterAnswersReputationLevels	ChatterAnswersReputationLevel []	Contains the name and value pair that describes the reputation level for Chatter Answers. Available in API version 28.0 and later.
ideaReputationLevels	IdeaReputationLevel	Contains the name and value pair that describes the reputation for Ideas. Available in API version 28.0 and later.

ChatterAnswersReputationLevel

Represents the reputation name and the number of points for that level for Chatter Answers.

Field Name	Field Type	Description
name	string	The name of the reputation level, for example, "Expert."
value	int	The minimum number of points for the reputation level.

IdeaReputationLevel

Represents the reputation name and the number of points for that level for Ideas. Available in API version 28.0 and later.

Field Name	Field Type	Description
name	string	The name of the reputation level, for example, "Expert."

Field Name	Field Type	Description
`value`	int	The minimum number of points for the reputation level.

Declarative Metadata Sample Definition

The following is the definition of a community (zone) component:

```
<?xml version="1.0" encoding="UTF-8"?>
<Community xmlns="http://soap.sforce.com/2006/04/metadata">
    <active>true</active>
    <communityFeedPage>communityWithHeaderAndFooter_main</communityFeedPage>
    <description>Metadata Test</description>
    <emailFooterDocument>sampleFolder/emailFooter.html</emailFooterDocument>
    <emailHeaderDocument>sampleFolder/emailHeader.html</emailHeaderDocument>
    <enableChatterAnswers>true</enableChatterAnswers>
    <enablePrivateQuestions>true</enablePrivateQuestions>
    <expertsGroup>CommunityExperts</expertsGroup>
    <portal>Customer Portal</portal>
    <emailNotificationUrl>http://yourURL</emailNotificationUrl>
<reputationLevels>
        <chatterAnswersReputationLevels>
            <name>Newbie</name>
            <value>0</value>
        </chatterAnswersReputationLevels>
        <chatterAnswersReputationLevels>
            <name>Smartie</name>
            <value>500</value>
        </chatterAnswersReputationLevels>
        <chatterAnswersReputationLevels>
            <name>Pro</name>
            <value>2000</value>
        </chatterAnswersReputationLevels>
        <chatterAnswersReputationLevels>
            <name>All Star</name>
            <value>5000</value>
        </chatterAnswersReputationLevels>
        <ideaReputationLevels>
            <name>Observer</name>
            <value>0</value>
        </ideaReputationLevels>
        <ideaReputationLevels>
            <name>Contributor</name>
            <value>100</value>
        </ideaReputationLevels>
        <ideaReputationLevels>
            <name>Influencer</name>
            <value>400</value>
        </ideaReputationLevels>
        <ideaReputationLevels>
            <name>Thought Leader</name>
            <value>1500</value>
        </ideaReputationLevels>
    </reputationLevels>
    <showInPortal>true</showInPortal>
    <site>ChatterAnswersSite</site>
</Community>
```

Wildcard Support in the Manifest File

This metadata type supports the wildcard character `*` (asterisk) in the `package.xml` manifest file. For information about using the manifest file, see Deploying and Retrieving Metadata with the Zip File.

CommunityTemplateDefinition

Represents the definition of a community template. This type extends the Metadata metadata type and inherits its `fullName` field.

File Suffix and Directory Location

CommunityTemplateDefinition components have the suffix `.communityTemplateDefinition` and are stored in the `communityTemplateDefinitions` folder.

Version

CommunityTemplateDefinition components are available in API version 38.0 and later.

Special Access Rules

This type is available only if Salesforce Communities is enabled in your org.

Fields

Field Name	Field Type	Description
`baseTemplate`	CommunityBaseTemplate (enumeration of type string)	Denotes that this CommunityTemplateDefinition was created in API version 41.0 or later. The only valid value is `c`. This field is available in API 41.0 and later.
`bundlesInfo`	CommunityTemplateBundleInfo []	The list of preview images and feature highlights of this CommunityTemplateDefinition.
`category`	CommunityTemplateCategory (enumeration of type string)	Required. The optimized use case of this CommunityTemplateDefinition. Valid values are: • `IT` • `Marketing` • `Sales` • `Service`

Field Name	Field Type	Description
defaultBrandingSet	string	The set of branding properties associated with this CommunityTemplateDefinition, as defined in the Branding panel in Experience Builder. Available in API version 40.0 and later. In API version 44.0 and later, this field is read-only and can be implemented in CommunityThemeDefinition on page 259.
defaultThemeDefinition	string	Required. The assigned theme definition for this CommunityTemplateDefinition.
description	string	The optional description text of this CommunityTemplateDefinition.
enableExtendedCleanUpOnDelete	boolean	False by default. Determines if deleting this CommunityTemplateDefinition attempts to delete other directly or indirectly referenced objects automatically, for example, CommunityThemeDefinition on page 259, Flexipage on page 517, or StaticResource on page 1069. Values are true or false.
masterLabel	string	Required. The label for this CommunityTemplateDefinition, which displays in Setup.
navigationLinkSet	NavigationLinkSet	The navigation menu associated with this CommunityTemplateDefinition. A navigation menu consists of items that users can click to go to other parts of the community. Available in API versions 37.0 to 46.0. In API versions 47.0 and later, use NavigationMenu.
pageSetting	CommunityTemplatePageSetting[]	The list of FlexiPages of this CommunityTemplateDefinition.
publisher	string	Defines the name of the publisher as seen in the Change Theme wizard. If no name is provided, the name of the org from which the package was originally exported is used. This field is available in API version 45.0 and later.

CommunityTemplateBundleInfo

Field Name	Field Type	Description
description	string	The optional description text of its CommunityTemplateBundleInfo.
image	string	Required only when the type is PreviewImage, otherwise this field is optional. A preview image for this CommunityTemplateDefinition.
order	int	Required. An integer specifying the position of this CommunityTemplateBundleInfo relative to others of the same type within its CommunityTemplateDefinition. 1 is the first position, 3 is the maximum position for PreviewImage type, and 4 is the maximum position for the Highlight type.
title	string	Required. The title of this CommunityTemplateBundleInfo to use in code.

Field Name	Field Type	Description
`type`	CommunityTemplate BundleInfoType (enumeration of type string)	Required. Stores descriptive information about the template that is included in the export. The template powers the interface of the Community Creation Wizard. Valid values are: • `Highlight`—This CommunityTemplateBundleInfo is used as a highlighted feature. Up to 4 are supported. • `PreviewImage`—This CommunityTemplateBundleInfo is used as a preview image. Up to 3 are supported.

CommunityTemplatePageSetting

Field Name	Field Type	Description
`page`	string	Required. The list of FlexiPages of this CommunityTemplateDefinition.
`themeLayout`	string	Required. The name of the FlexiPage for the theme layout. This field is available in API version 39.0 and later.

Declarative Metadata Sample Definition

The following is an example of a CommunityTemplateDefinition component.

```
<?xml version="1.0" encoding="UTF-8"?>
<CommunityTemplateDefinition xmlns="http://soap.sforce.com/2006/04/metadata">
    <baseTemplate>c</baseTemplate>
    <bundlesInfo>
        <description>Feature Description</description>
        <order>1</order>
        <title>Feature Heading</title>
        <type>Highlight</type>
    </bundlesInfo>
    <bundlesInfo>
        <image>siteAsset_2dbe594eb6794173af78da264cd6a4a7</image>
        <order>1</order>
        <title>Preview Image</title>
        <type>PreviewImage</type>
    </bundlesInfo>
    <category>Sales</category>
    <defaultThemeDefinition>communityTemplate</defaultThemeDefinition>
    <description>This is a lightning community template</description>
    <enableExtendedCleanUpOnDelete>true</enableExtendedCleanUpOnDelete>
    <masterLabel>communityTemplate</masterLabel>
    <navigationLinkSet>
        <navigationMenuItem>
            <label>Topics</label>
            <position>0</position>
            <publiclyAvailable>true</publiclyAvailable>
            <target>ShowMoreTopics</target>
            <type>NavigationalTopic</type>
        </navigationMenuItem>
    </navigationLinkSet>
    <pageSetting>
        <page>communityTemplate_Report_List</page>
        <themeLayout>communityTemplate_themeLayout_Default</themeLayout>
    </pageSetting>
    <pageSetting>
        <page>communityTemplate_Topic_Catalog</page>
        <themeLayout>communityTemplate_themeLayout_Default</themeLayout>
    </pageSetting>
    <pageSetting>
        <page>communityTemplate_Check_Password</page>
        <themeLayout>communityTemplate_themeLayout_Login</themeLayout>
    </pageSetting>
    <pageSetting>
        <page>communityTemplate_Error</page>
        <themeLayout>communityTemplate_themeLayout_Default</themeLayout>
    </pageSetting>
    <pageSetting>
        <page>communityTemplate_User_Settings</page>
        <themeLayout>communityTemplate_themeLayout_Default</themeLayout>
    </pageSetting>
    <pageSetting>
        <page>communityTemplate_Login</page>
        <themeLayout>communityTemplate_themeLayout_Login</themeLayout>
    </pageSetting>
    <pageSetting>
        <page>communityTemplate_Stream_List</page>
        <themeLayout>communityTemplate_themeLayout_Default</themeLayout>
    </pageSetting>
    <pageSetting>
        <page>communityTemplate_Sfdc_Page</page>
        <themeLayout>communityTemplate_themeLayout_Default</themeLayout>
    </pageSetting>
    <pageSetting>
        <page>communityTemplate_Group_Detail</page>
        <themeLayout>communityTemplate_themeLayout_Default</themeLayout>
    </pageSetting>
    <pageSetting>
        <page>communityTemplate_Report_Related_List</page>
        <themeLayout>communityTemplate_themeLayout_Default</themeLayout>
    </pageSetting>
    <pageSetting>
        <page>communityTemplate_Register</page>
        <themeLayout>communityTemplate_themeLayout_Login</themeLayout>
    </pageSetting>
    <pageSetting>
        <page>communityTemplate_User_Profile</page>
        <themeLayout>communityTemplate_themeLayout_Default</themeLayout>
    </pageSetting>
    <pageSetting>
        <page>communityTemplate_Case_Detail</page>
        <themeLayout>communityTemplate_themeLayout_Default</themeLayout>
    </pageSetting>
    <pageSetting>
        <page>communityTemplate_Stream_Related_List</page>
        <themeLayout>communityTemplate_themeLayout_Default</themeLayout>
    </pageSetting>
    <pageSetting>
        <page>communityTemplate_Dashboard_Detail</page>
        <themeLayout>communityTemplate_themeLayout_Default</themeLayout>
    </pageSetting>
    <pageSetting>
        <page>communityTemplate_Group_List</page>
        <themeLayout>communityTemplate_themeLayout_Default</themeLayout>
    </pageSetting>
    <pageSetting>
        <page>communityTemplate_Canvasapp_Page</page>
        <themeLayout>communityTemplate_themeLayout_Default</themeLayout>
    </pageSetting>
    <pageSetting>
        <page>communityTemplate_Login_Error</page>
        <themeLayout>communityTemplate_themeLayout_Login</themeLayout>
    </pageSetting>
    <pageSetting>
        <page>communityTemplate_Create_Record</page>
        <themeLayout>communityTemplate_themeLayout_Default</themeLayout>
    </pageSetting>
    <pageSetting>
        <page>communityTemplate_Group_Related_List</page>
        <themeLayout>communityTemplate_themeLayout_Default</themeLayout>
    </pageSetting>
    <pageSetting>
        <page>communityTemplate_Search</page>
        <themeLayout>communityTemplate_themeLayout_Default</themeLayout>
    </pageSetting>
    <pageSetting>
        <page>communityTemplate_File_Detail</page>
        <themeLayout>communityTemplate_themeLayout_Default</themeLayout>
    </pageSetting>
    <pageSetting>
        <page>communityTemplate_Case_List</page>
        <themeLayout>communityTemplate_themeLayout_Default</themeLayout>
    </pageSetting>
    <pageSetting>
        <page>communityTemplate_User_List</page>
        <themeLayout>communityTemplate_themeLayout_Default</themeLayout>
    </pageSetting>
    <pageSetting>
        <page>communityTemplate_File_List</page>
        <themeLayout>communityTemplate_themeLayout_Default</themeLayout>
    </pageSetting>
    <pageSetting>
        <page>communityTemplate_Question_Detail</page>
        <themeLayout>communityTemplate_themeLayout_Default</themeLayout>
    </pageSetting>
    <pageSetting>
```

```
        <page>communityTemplate_Dashboard_List</page>
        <themeLayout>communityTemplate_themeLayout_Default</themeLayout>
    </pageSetting>
    <pageSetting>
        <page>communityTemplate_Related_Record_List</page>
        <themeLayout>communityTemplate_themeLayout_Default</themeLayout>
    </pageSetting>
    <pageSetting>
        <page>communityTemplate_File_Related_List</page>
        <themeLayout>communityTemplate_themeLayout_Default</themeLayout>
    </pageSetting>
    <pageSetting>
        <page>communityTemplate_Record_List</page>
        <themeLayout>communityTemplate_themeLayout_Default</themeLayout>
    </pageSetting>
    <pageSetting>
        <page>communityTemplate_Forgot_Password</page>
        <themeLayout>communityTemplate_themeLayout_Login</themeLayout>
    </pageSetting>
    <pageSetting>
        <page>communityTemplate_Home</page>
        <themeLayout>communityTemplate_themeLayout_Default</themeLayout>
    </pageSetting>
    <pageSetting>
        <page>communityTemplate_Dashboard_Related_List</page>
        <themeLayout>communityTemplate_themeLayout_Default</themeLayout>
    </pageSetting>
    <pageSetting>
        <page>communityTemplate_Account_Management</page>
        <themeLayout>communityTemplate_themeLayout_Default</themeLayout>
    </pageSetting>
    <pageSetting>
        <page>communityTemplate_Case_Related_List</page>
        <themeLayout>communityTemplate_themeLayout_Default</themeLayout>
    </pageSetting>
    <pageSetting>
        <page>communityTemplate_User_Related_List</page>
        <themeLayout>communityTemplate_themeLayout_Default</themeLayout>
    </pageSetting>
    <pageSetting>
        <page>communityTemplate_Stream_Detail</page>
        <themeLayout>communityTemplate_themeLayout_Default</themeLayout>
    </pageSetting>
    <pageSetting>
        <page>communityTemplate_Topic_Detail</page>
        <themeLayout>communityTemplate_themeLayout_Default</themeLayout>
    </pageSetting>
    <pageSetting>
        <page>communityTemplate_Messages</page>
        <themeLayout>communityTemplate_themeLayout_Default</themeLayout>
    </pageSetting>
    <pageSetting>
        <page>communityTemplate_Report_Detail</page>
        <themeLayout>communityTemplate_themeLayout_Default</themeLayout>
    </pageSetting>
    <pageSetting>
        <page>communityTemplate_Record_Detail</page>
        <themeLayout>communityTemplate_themeLayout_Default</themeLayout>
    </pageSetting>
    <pageSetting>
        <page>communityTemplate_Feed_Detail</page>
        <themeLayout>communityTemplate_themeLayout_Default</themeLayout>
    </pageSetting>
    <pageSetting>
        <page>communityTemplate_Contact_Support</page>
        <themeLayout>communityTemplate_themeLayout_Default</themeLayout>
    </pageSetting>
</CommunityTemplateDefinition>
```

The following is an example `package.xml` that references the previous definition.

```
<?xml version="1.0" encoding="UTF-8"?>
<Package xmlns="http://soap.sforce.com/2006/04/metadata">
    <types>
        <members>MyTemplate</members>
        <name>CommunityTemplateDefinition</name>
    </types>
    <version>48.0</version>
</Package>
```

Wildcard Support in the Manifest File

This metadata type supports the wildcard character `*` (asterisk) in the `package.xml` manifest file. For information about using the manifest file, see Deploying and Retrieving Metadata with the Zip File.

CommunityThemeDefinition

Represents the definition of a community theme.This type extends the Metadata metadata type and inherits its `fullName` field.

File Suffix and Directory Location

CommunityThemeDefinition components have the suffix `.communityThemeDefinition` and are stored in the `communityThemeDefinitions` folder.

Version

CommunityThemeDefinition components are available in API version 38.0 and later.

Special Access Rules

This type is available only if Salesforce Communities is enabled in your org.

Fields

Field Name	Field Type	Description
bundlesInfo	CommunityThemeBundleInfo []	If specified, at least one preview image and one highlight are required. Up to 3 preview images and 4 highlights are supported. Available in API version 44.0 and later
customThemeLayoutType	CommunityCustomThemeLayoutType []	The list of custom theme layout types available to the theme layout.
defaultBrandingSet	string	The set of branding properties associated with this CommunityThemeDefinition, as defined in the Branding panel in Experience Builder. Available in API version 44.0 and later.
description	string	The optional description text of this CommunityThemeDefinition.
enableExtendedCleanUpOnDelete	boolean	False by default. Determines if deleting this CommunityThemeDefinition attempts to delete other directly or indirectly referenced objects automatically, for example, FlexiPage. Values are true or false.
masterLabel	string	Required. The label for this CommunityThemeDefinition, which displays in Setup.
publisher	string	Defines the name of the publisher as seen in the Community Creation wizard. If no name is provided, the name of the org from which the package was originally exported is used. This field is available in API version 45.0 and later.
themeRouteOverride	CommunityThemeRouteOverride []	List of theme layout type overrides for flexipages (currently only for home). Available in API version 44.0 and later.
themeSetting	CommunityThemeSetting []	Required. The list of settings for this CommunityThemeDefinition.

CommunityThemeBundleInfo

Field Name	Field Type	Description
description	string	The optional description text of its CommunityThemeBundleInfo.
image	string	Required only when the type is PreviewImage, otherwise this field is optional. A preview image for this CommunityThemeDefinition.
order	int	Required. An integer specifying the position of this CommunityThemeBundleInfo relative to others of the same type within its CommunityThemeDefinition. 1 is the first position, 3 is the maximum position for PreviewImage type, and 4 is the maximum position for the Highlight type.
title	string	Required. The title of this CommunityThemeBundleInfo to use in code.

Field Name	Field Type	Description
type	CommunityTemplate BundleInfoType (enumeration of type string)	Required. Stores descriptive information about the theme that is included in the export. Valid values are: • Highlight—This CommunityThemeBundleInfo is used as a highlighted feature. Up to 4 are supported. • PreviewImage—This CommunityThemeBundleInfo is used as a preview image. Up to 3 are supported.

CommunityCustomThemeLayoutType

Field Name	Field Type	Description
description	string	The description of the custom theme layout type.
label	string	Required. The name of the custom theme layout type. The values Inner, Home, and Login are reserved.

CommunityThemeRouteOverride

Field Name	Field Type	Description
customThemeLayoutType	string	Required when themeLayoutType is not specified. Provides the custom theme layout type associated with the theme layout. This field and themeLayoutType are mutually exclusive; you can't specify both.
pageAttributes	string	Required. Specifies the attributes of the community page for which the default theme layout type is overridden. The only valid value is {"PageName":"Home"}.
pageType	string	Required. Specifies the type of the community page for which the default theme layout type is overridden. The only valid value is comm__standardPage.
themeLayoutType	CommunityTheme LayoutType (enumeration of type string)	Required if customThemeLayoutType is not specified. Provides the default theme layout type associated with the theme layout. Valid values are Inner, Home, or Login. This field and customThemeLayoutType are mutually exclusive; you can't specify both.

CommunityThemeSetting

Field Name	Field Type	Description
customThemeLayoutType	string	Required when themeLayoutType is not specified. The custom theme layout type associated with the theme layout. This field and themeLayoutType are mutually exclusive; you can't specify both.

Field Name	Field Type	Description
`themeLayout`	string	Required. The configuration and layout for this theme.
`themeLayoutType`	CommunityThemeLayoutType (enumeration of type string)	Required when `customThemeLayoutType` is not specified. The default theme layout type associated with the theme layout. Valid values are `Inner`, `Home`, or `Login`. This field and `customThemeLayoutType` are mutually exclusive; you can't specify both.

Declarative Metadata Sample Definition

The following is an example of a CommunityThemeDefinition component.

```
<?xml version="1.0" encoding="UTF-8"?>
<CommunityThemeDefinition xmlns="http://soap.sforce.com/2006/04/metadata">
    <bundlesInfo>
        <description>Batman Feature1 description</description>
        <order>1</order>
        <title>Batman Feature1</title>
        <type>Highlight</type>
    </bundlesInfo>
    <bundlesInfo>
        <image>siteAsset_d90e2d5ce4cf4d8899e233c051091246</image>
        <order>1</order>
        <title>siteAsset_d90e2d5ce4cf4d8899e233c051091246</title>
        <type>PreviewImage</type>
    </bundlesInfo>
    <defaultBrandingSet>Batman</defaultBrandingSet>
    <description>Batman theme</description>
    <enableExtendedCleanUpOnDelete>true</enableExtendedCleanUpOnDelete>
    <masterLabel>Batman</masterLabel>
    <themeRouteOverride>
        <pageAttributes>{&quot;PageName&quot;:&quot;Home&quot;}</pageAttributes>
        <pageType>comm__standardPage</pageType>
        <themeLayoutType>Home</themeLayoutType>
    </themeRouteOverride>
    <themeSetting>
        <themeLayout>Batman_themeLayout_Login</themeLayout>
        <themeLayoutType>Login</themeLayoutType>
    </themeSetting>
    <themeSetting>
        <themeLayout>Batman_themeLayout_Home</themeLayout>
        <themeLayoutType>Home</themeLayoutType>
    </themeSetting>
    <themeSetting>
        <themeLayout>Batman_themeLayout_Default</themeLayout>
        <themeLayoutType>Inner</themeLayoutType>
    </themeSetting>
</CommunityThemeDefinition>
```

The following is an example `package.xml` that references the previous definition.

```
<?xml version="1.0" encoding="UTF-8"?>
<Package xmlns="http://soap.sforce.com/2006/04/metadata">
```

```
    <types>
        <members>Batman</members>
        <name>BrandingSet</name>
    </types>
    <types>
        <members>Batman</members>
        <name>CommunityThemeDefinition</name>
    </types>
    <types>
        <members>Batman_themeLayout_Default</members>
        <members>Batman_themeLayout_Home</members>
        <members>Batman_themeLayout_Login</members>
        <name>FlexiPage</name>
    </types>
    <types>
        <members>siteAsset_d90e2d5ce4cf4d8899e233c051091246</members>
        <name>StaticResource</name>
    </types>
    <version>48.0</version>
</Package>
```

Wildcard Support in the Manifest File

This metadata type supports the wildcard character `*` (asterisk) in the `package.xml` manifest file. For information about using the manifest file, see Deploying and Retrieving Metadata with the Zip File.

ConnectedApp

Represents a connected app configuration. A connected app integrates an application with Salesforce using APIs. Connected apps use standard SAML and OAuth protocols to authenticate, provide single sign-on, and provide tokens for use with Salesforce APIs. In addition to standard OAuth capabilities, Salesforce admins set various security policies for connected apps and have explicit control over who can use the corresponding apps.

This type extends the Metadata metadata type and inherits its `fullName` field.

File Suffix and Directory Location

ConnectedApp components have the suffix `.connectedApp` and are stored in the `connectedApps` folder.

Version

ConnectedApp components are available in API version 29.0 and later.

Fields

Field Name	Field Type	Description
`attributes`	ConnectedAppAttribute	A custom attribute of the connected app.

Field Name	Field Type	Description
canvasConfig	ConnectedAppCanvasConfig	The configuration options of the connected app if it's exposed as a canvas app.
contactEmail	string	Required. The email address Salesforce uses for contacting you or your support team.
contactPhone	string	The phone number for Salesforce to use to contact you.
description	string	An optional description for your application.
iconUrl	string	Reserved for future use.
infoUrl	string	An optional URL for a web page with more information about your application.
ipRanges	ConnectedAppIpRange	Specifies the ranges of IP addresses that can access the app without requiring the user to authenticate with the connected app.
label	string	Required. The name of the app.
logoUrl	string	An optional application logo. The logo appears with the application's entry in the list of apps and on the consent page the user sees when authenticating. The URL must use HTTPS, and the logo can't be larger than 125 pixels high or 200 pixels wide. The default logo is a cloud.
mobileStartUrl	string	Users are directed to this URL after they've authenticated when the app is accessed from a mobile device. If you don't give a URL, the user is sent to the application's default start page after authentication completes. If the connected app that you're creating is a canvas app, then you can leave this field blank. The Canvas App URL field contains the URL that gets called for the connected app.
oauthConfig	ConnectedAppOauthConfig	Specifies how your application communicates with Salesforce.
permissionSetName	string	Specifies the permissions required to perform different functions with the connected app. Available in API version 46.0 and later. You can assign multiple permission sets to the connected app, but you must enter each permission set name on a separate line. You can't enter the same permission set name more than one time for each connected app. You can also change a permission set by replacing the current permission set with a new permission set. Make sure that each permission set name assigned to the connected app is unique. You can delete individual permission sets or remove all permission sets from a connected app by entering an empty permissionSetName string on deployment of the connected app (<permissionSetName></permissionSetName>). Note: To use this field, the isAdminApproved field on the ConnectedAppOauthConfig subtype must be set to true.

Field Name	Field Type	Description
`plugin`	string	The name of a custom Apex class that extends `Auth.ConnectedAppPlugin` to customize the behavior of the app.
`pluginExecutionUser`	string	Specifies the user to run the plugin as. If the user isn't authorized to use the connected app, use the `authorize` method. See the `ConnectedAppPlugin` class in the Apex Developer Guide. Available in API version 46.0 and later. Enter a user that is part of your org. Otherwise, the user is removed from this field when you deploy the connected app. If you don't want to specify a user, you can leave this field empty. Note: To use this field in an org, the ConAppPluginExecuteAsUser org perm must be enabled.
`profileName`	string	Specifies the profile (base-level user permissions) required to perform different functions with the connected app. Available in API version 46.0 and later. You can assign multiple profiles to the connected app, but you must enter each profile name on a separate line. You can't enter the same profile name more than one time for each connected app. You can also change profiles by replacing the current profiles with new profiles. Make sure that each profile name assigned to the connected app is unique. You can also delete individual profiles, or remove all profiles from a connected app by entering an empty `profileName` string on deployment of the connected app (`<profileName></profileName>`). Note: To use this field, the `isAdminApproved` field on the ConnectedAppOauthConfig subtype must be set to `true`.
`samlConfig`	ConnectedAppSamlConfig	Controls how the app uses single sign-on.
`startUrl`	string	If the app is not accessed from a mobile device, users are directed to this URL after they've authenticated. If you don't give a URL, the user is sent to the application's default start page after authentication completes. If the app is accessed from a mobile device, see `mobileStartUrl`. If the connected app that you're creating is a canvas app, then you can leave this field empty. The Canvas App URL field contains the URL that gets called for the connected app.

ConnectedAppAttribute

Represents the field names that make up a custom attribute when using SAML with a ConnectedApp. Tailor these values to a specific service provider.

Field Name	Field Type	Description
`formula`	string	Required. The value of the attribute.
`key`	string	Required. The attribute's identifier.

ConnectedAppCanvasConfig

The configuration options of the connected app if it's exposed as a canvas app.

Field Name	Field Type	Description
`accessMethod`	AccessMethod (enumeration of type string)	Required. Indicates how the canvas app initiates the OAuth authentication flow. The valid values are: • `Get`—OAuth authentication is used, and the user is prompted to allow the third-party application to access their information. When you use this access method, the canvas app must initiate the OAuth authentication flow. • `Post`—OAuth authentication is used, but when the administrator installs the canvas app, they implicitly allow access for users. Therefore, the user isn't prompted to allow the third party to access their user information. When you use this access method, the authentication is posted directly to the canvas app URL.
`canvasUrl`	string	Required. The URL of the third-party app that's exposed as a canvas app.
`lifecycleClass`	string	The name of the `Canvas.CanvasLifecycleHandler` Apex class, if you've implemented this class for custom parameters. This field is available in API version 31.0 and later.
`locations`	CanvasLocationOptions (enumeration of type string)	Indicates where the canvas app can appear to the user. The valid values are: • `Aura`—Reserved for future use. • `AppLauncher`—Reserved for future use. • `Chatter`—The canvas app can appear in the app navigation list on the Chatter tab. • `ChatterFeed`—The canvas app can appear as a Chatter feed item. • `MobileNav`—The canvas app can appear in a mobile card in the Salesforce app. This value is available in API version 31.0 and later. • `None`—The canvas app can appear only in the Canvas App Previewer. • `OpenCTI`—The canvas app can appear in the call control tool. • `PageLayout`—The canvas app can appear on a page layout. When viewed in the Salesforce app, the canvas app appears in the

Field Name	Field Type	Description
		record detail page. This value is available in API version 31.0 and later. - `Publisher`—The canvas app can appear as a global action. - `ServiceDesk`—The canvas app can appear in the footer or sidebars of a Salesforce console. - `UserProfile`—Reserved for future use. - `Visualforce`—The canvas app can appear on a Visualforce page.
`options`	CanvasOptions (enumeration of type string)	Indicates whether to hide the share button and header in the publisher for your canvas app, and whether the app is a canvas personal app. Valid values are: - `HideShare`—The Share button is hidden in the publisher for the related canvas app. Available in API version 30.0 and later. - `HideHeader`—The header is hidden in the publisher for the related canvas app. Available in API version 30.0 and later. - `PersonalEnabled`—End users can install the app as a canvas personal app. Available in API version 32.0 and later.
`samlInitiationMethod`	SamlInitiationMethod (enumeration of type string)	If you're using SAML single sign-on (SSO), indicates which provider initiates the SSO flow. - `IdpInitiated`—Identity provider initiated. Salesforce makes the initial request to start the SSO flow. - `SpInitiated`—Service provider initiated. The canvas app starts the SSO flow after it's invoked. - `None`—The canvas app isn't using SAML SSO. This field is available in API version 31.0 and later.

ConnectedAppIpRange

The list of IP addresses that can access the app without requiring the user to authenticate.

Field Name	Field Type	Description
`description`	string	Use this field to identify the purpose of the range, such as which part of a network corresponds to this range. This field is available in API version 31.0 and later.
`startAddress`	string	Required. The first address in the IP range, inclusive.

Field Name	Field Type	Description
`endAddress`	string	Required. The last address in the IP range, inclusive.

ConnectedAppOauthConfig

Represents the field names that configure how your application communicates with Salesforce.

Field Name	Field Type	Description
`callbackUrl`	string	Required. The endpoint that Salesforce calls back to your application during OAuth; it's the OAuth `redirect_uri`.
`certificate`	string	The PEM-encoded certificate string, if the app uses a certificate.
`consumerKey`	string	A value used by the consumer for identification to Salesforce. Referred to as `client_id` in OAuth 2.0. In API version 32.0 and later, you can set this field's value only during creation. After you define and save the value, it can't be edited. The value must be alphanumeric, can't contain special characters or spaces, and must be between 8 and 256 characters. Consumer keys must be globally unique.
`consumerSecret`	string	A value that is combined with the `consumerKey` and used by the consumer for identification to Salesforce. Referred to as `client_secret` in OAuth 2.0. Typically, Salesforce generates this value when you create the connected app. However, you can customize the shared secret value during creation. After you save the value, it can't be edited. When set, the value is not returned in metadata API requests. The value must be alphanumeric (no special characters and no spaces) and a minimum of 8 characters (maximum of 256 characters). If you specify a secret already in use for another connected app in the organization, an error occurs. This field is available in API version 32.0 and later.
`idTokenConfig`	ConnectedAppOauthIdToken	Specifies the ID token configuration for the connected app OAuth settings. Available in API version 43.0 and later.
`isAdminApproved`	boolean	If set to `false` (default setting), anyone in the org can authorize the app. Users must approve the app the first time they access it. If set to `true`, only users with the appropriate profile or permission set can access the app. These users don't have to approve the app before they can access it. Manage profiles for the app by editing each profile's Connected App Access list. Manage permission sets for the app by editing each permission set's Assigned Connected App list. This setting is not available in Group Edition. Available in API version 46.0 and later.

Field Name	Field Type	Description
		Note: Connected app consumers can edit this setting when deploying a connected app in their org.
`scopes`	ConnectedAppOauthAccessScope (enumeration of type string)	The scopes refer to permissions given by the user running the connected app. When deploying metadata, valid values are: • `Address`—Allows access to the logged-in user's street address (the same behavior as deploying `Basic`). • `Api`—Allows access to the logged-in user's account over the APIs. • `Basic`—Allows access to your identity URL service (the same behavior as deploying `Address`, `Email`, `Phone`, and `Profile`). • `Chatter`—Allows access to only the Chatter REST API resources. • `CustomApplications`—Provides access to custom applications, such as those using Visualforce. • `CustomPermissions`—Allows access to the custom permissions in an organization associated with the connected app, and shows whether the current user has each permission enabled. • `Email`—Allows access to the logged-in user's email address (the same behavior as deploying `Basic`). • `Full`—Allows access to all data accessible by the logged-in user. • `OfflineAccess`—Allows the app to interact with the user's data while the user is offline, and get a refresh token (the same behavior as deploying `RefreshToken`). • `OpenID`—Allows access to the logged-in user's unique identifier for OpenID Connect apps. • `Phone`—Allows access to the logged-in user's phone number value (the same behavior as deploying `Basic`). • `Profile`—Allows access to the logged-in user's profile (the same behavior as deploying `Basic`). • `RefreshToken`—Allows a refresh token to be returned if you're eligible to receive one (the same behavior as deploying `OfflineAccess`). • `Wave`—Allows access to the Analytics REST API resources. Available in API version 35.0 and later. • `Web`—Allows use of the `access_token` on the web. This usage also includes `visualforce`, allowing access to Visualforce pages. When retrieving metadata, valid values are: • `Api`—Allows access to the logged-in user's account over the APIs. • `Basic`—Allows access to the user's identity URL service, and includes `Address`, `Email`, `Phone`, and `Profile`. • `Chatter`—Allows access to only the Chatter REST API resources.

Field Name	Field Type	Description
		- `CustomApplications`—Allows access to custom applications, such as those using Visualforce. - `Full`—Allows access to all data accessible by the logged-in user. - `OpenID`—Allows access to the logged in user's unique identifier for OpenID Connect apps. - `RefreshToken`—Allows a refresh token to be returned if you are eligible to receive one, and is synonymous with allowing `OfflineAccess`. - `Wave`—Allows access to the Analytics REST API resources. Available in API version 35.0 and later. - `Web`—Allows usage of the `access_token` on the web. This usage also includes `visualforce`, allowing access to Visualforce pages.
`singleLogoutUrl`	string	The single logout endpoint. This URL is the endpoint where Salesforce sends a logout request when users log out of Salesforce.

ConnectedAppOauthIdToken

Specifies the ID token configuration for the connected app OAuth settings. Available in API version 43.0 and later.

Field Name	Field Type	Description
`idTokenAudience`	string	The audiences that this ID token is intended for. The value is an array of case-sensitive strings. If no audiences are specified, the OAuth 2.0 `client_id` of the relying party is returned as the default audience. Otherwise, the other audiences are returned with the `client_id` in the `aud` value.
`idTokenIncludeAttributes`	boolean	Indicates whether attributes are included in the ID token.
`idTokenIncludeCustomPerms`	boolean	Indicates whether custom permissions are included in the ID token.
`idTokenIncludeStandardClaims`	boolean	Indicates whether standard claims about the authentication event are included in the ID token.
`idTokenValidity`	int	The length of time that the ID token is valid for after it's issued. The value can be from 1 to 720 minutes. The default is 2 minutes.

ConnectedAppSamlConfig

Specifies how an app uses single sign-on.

Field Name	Field Type	Description
`acsUrl`	string	Required. The assertion consumer service URL from the service provider.

Field Name	Field Type	Description
`certificate`	string	The PEM-encoded certificate string, if the app uses a certificate.
`entityUrl`	string	Required. The entity ID from your service provider.
`encryptionCertificate`	string	The name of the certificate to use for encrypting SAML assertions to the service provider. This certificate is saved in the organization's Certificate and Key Management list. Available in API version 30.0 and later .
`encryptionType`	SamlEncryptionType (enumeration of type string)	When Salesforce is the identity provider, the SAML configuration can specify the encryption method used for encrypting SAML assertions to the service provider. The service provider detects the encryption method in the SAML assertion for decryption. Valid values are: • `AES_128`—128–bit key. • `AES_256`—256–bit key. • `Triple_Des`—Triple Data Encryption Algorithm. Available in API version 30.0 and later.
`issuer`	string	A URI that sends the SAML response. A service provider can use this URI to determine which identity provider sent the response. Available in API version 29.0 and later.
`samlIdpSLOBinding`	SamlIdpSLOBinding (enumeration of type string)	The SAML HTTP binding type from the service provider used for single logout. Available in API version 40.0 and later. Valid values are: • `PostBinding` • `RedirectBinding`
`samlNameIdFormat`	SamlNameIdFormatType (enumeration of type string)	Indicates the format the service provider (SP) requires for the user's single sign-on identifier. Available in API version 29.0 and later. Valid values are: • `Unspecified`—No format given. This option is the default. • `EmailAddress`—Used if the subject type is the user's name ora federation ID (an ID internal to the SP). • `Persistent`—Used with the user ID and persistent ID subject types. • `Transient`—Used when the subject type is a custom attribute and can change every time the user logs in.
`samlSloUrl`	string	The SAML single-logout endpoint of the connected app service provider (SP). This endpoint is where SAML LogoutRequests and LogoutResponses are sent when users log out of Salesforce. The SP provides this endpoint. Available in API version 40.0 and later.
`samlSubjectCustomAttr`	string	If the `samlSubjectType` is `CustomAttr`, include that custom value here; otherwise, leave empty. Available in API version 29.0 and later.

Field Name	Field Type	Description
samlSubjectType	SamlSubjectType (enumeration of type string)	Required. The single sign-on identifier for the user. Valid values are: • Username—The user's Salesforce name. • FederationId—The user's identifier at the service provider. Get this value from the service provider. • UserId—The user's Salesforce identifier. • PersistentID—A persistent opaque identifier that is specific to the identity provider and a service provider. • CustomAttr—The identifier is taken from a custom field value in samlSubjectCustomAttr.

Declarative Metadata Sample Definition

The following is an example package manifest used to deploy or retrieve the ConnectedApp metadata for an organization.

```
<?xml version="1.0" encoding="UTF-8"?>
<Package xmlns="http://soap.sforce.com/2006/04/metadata">
    <types>
        <members>PortalTestApp</members>
        <name>ConnectedApp</name>
    </types>
    <version>29.0</version>
</Package>
```

The following is an example of a ConnectedApp component.

```
<?xml version="1.0" encoding="UTF-8"?>
<ConnectedApp xmlns="http://soap.sforce.com/2006/04/metadata">
    <fullName>AConnectedApp</fullName>
    <attributes>
        <formula>$User.CompanyName</formula>
        <key>companyName</key>
    </attributes>
    <contactEmail>user@company.com</contactEmail>
    <mobileStartUrl>https://m.connectedapp.company.com</mobileStartUrl> <label>A
ConnectedApp</label>
    <oauthConfig>
        <callbackUrl>https://callback.yourcompany.com</callbackUrl>
        <consumerKey>***********************************************</consumerKey>
        <isAdminApproved>true</isAdminApproved>
        <scopes>Basic</scopes>
        <scopes>Api</scopes>
        <scopes>Web</scopes>
        <scopes>Full</scopes>
        <scopes>Chatter</scopes>
        <scopes>CustomApplications</scopes>
        <scopes>RefreshToken</scopes>
        <scopes>OpenID</scopes>
        <scopes>CustomPermissions</scopes>
        <scopes>Wave</scopes>
        <scopes>Eclair</scopes>
    </oauthConfig>
    <permissionSetName>Advanced</permissionSetName>
    <permissionSetName>Service</permissionSetName>
    <profileName>Solution Manager</profileName>
    <profileName>System Administrator</profileName>
    <plugin>ConnectedAppPluginExample</plugin>
    <pluginExecutionUser>admin@salesforce.org</pluginExecutionUser>
    <startUrl>https://connectedapp.company.com</startUrl>
    <ipRanges>
        <endAddress>10.x.x.xx</endAddress>
        <startAddress>10.x.x.yy</startAddress>
    </ipRanges>
</ConnectedApp>
```

Usage

If you're constructing a SAML-enabled connected app using Metadata API, and must set the `IdP-Initiated Login URL` for your service provider, you have two options:

You can use the service provider app ID with the `app` parameter in the following format. This value is displayed in the Salesforce user interface. From Setup, enter *`Connected Apps`* in the Quick Find box, then select **Connected Apps**, then click the name of the connected app to see its detail page.

```
https://<Salesforce_base_URL>/idp/login?app=<app_id>
```

Or, if you're configuring the connected app using Metadata API only, you can use the `apiName` parameter of the service provider app in the following format. The `apiName` parameter is the `fullName` inherited from the Metadata type.

```
https://<Salesforce_base_URL>/idp/login?apiName=<fullName>
```

Wildcard Support in the Manifest File

This metadata type supports the wildcard character `*` (asterisk) in the `package.xml` manifest file. For information about using the manifest file, see Deploying and Retrieving Metadata with the Zip File.

ContentAsset

Represents the metadata for creating an asset file. Asset files enable a Salesforce file to be used for org setup and configuration purposes. This type extends the MetadataWithContent metadata type and inherits its `content` and `fullName` fields.

File Suffix and Directory Location

ContentAsset components have the suffix `.asset` and are stored in the `contentassets` folder.

Version

ContentAsset components are available in API version 38.0 and later.

Special Access Rules

The system prevents metadata retrieval if the total size of the asset's file content exceeds 30 MB. All pre-existing limits for packaging apply to asset files.

Fields

Field Name	Field Type	Description
`format`	ContentAssetFormat (enumeration of type string)	Describes the format of the asset file. Valid values are: • `Original`—A single asset file version. • `ZippedVersions`—Contains multiple versions of the asset file.
`isVisibleByExternalUsers`	boolean	Indicates whether unauthenticated users can see the asset file (`true`) or not (`false`). If not specified, the default value is `false`. This field is available in API version 44.0 and later.
`language`	string	Required. The language of the asset file label.
`masterLabel`	string	Required. The label for the asset file record, which displays in Setup.
`originNetwork`	string	For deploys, the name of the community the file is assigned upon creation. For retrievals, the name of the community the file is assigned to populates the field value. If `null`, file was not assigned to a community.
`relationships`	ContentAsset Relationships[]	The list of ContentAssetLinks that describe whether the asset file should be shared with the org.
`versions`	ContentAssetVersions	Required. Captures basic information about the file version(s) included the asset metadata. Typically the file has only one version.

ContentAssetRelationships

Represents the relationships between an asset file and the locations it's linked with.

Field Name	Field Type	Description
`organization`	ContentAssetLink[]	Carries information about sharing the asset file with the org. Maps to ContentDocumentLink.

ContentAssetLink

Represents a relationship link for an asset file, and includes details about the level of access for the link.

Field Name	Field Type	Description
access	ContentAssetAccess (enumeration of type string)	Required. The permission granted to the user of the shared file, determined by the permission the user already has. Valid values are: - VIEWER - COLLABORATOR - INFERRED
name	string	Reserved for future use.

ContentAssetVersions

Represents information about all file versions included in the asset metadata.

Field Name	Field Type	Description
version	ContentAssetVersion[]	A list of file versions for the asset.

ContentAssetVersion

Represents information about one file version included in the asset metadata.

Field Name	Field Type	Description
number	string	Required. The version number. This field is based on, or sets, the ContentVersion.
pathOnClient	string	Required. Describes the original filename of the file. This field maps to ContentVersion.PathOnClient. It provides the data for the ContentVersion Title field.
zipEntry	string	If the asset file has more than one version, format is ZippedVersions. In this case, zipEntry is the name of the file within the zip. If the asset file has only one version, this field is empty.

Declarative Metadata Sample Definition

The following is an example of a ContentAsset component.

```
<?xml version="1.0" encoding="UTF-8"?>
<ContentAsset xmlns="http://soap.sforce.com/2006/04/metadata">
    <masterLabel>some asset</masterLabel>
    <relationships>
        <organization>
            <access>VIEWER</access>
        </organization>
    </relationships>
    <versions>
        <version>
            <number>1</number>
            <pathOnClient>some asset.txt</pathOnClient>
        </version>
    </versions>
</ContentAsset>
```

For assets that include just one version, the format field can be omitted or specified with the value as `Original`. File assets with more than one version have versions wrapped in a zip file.

The following is an example `package.xml` that references the previous definition.

```
<?xml version="1.0" encoding="UTF-8"?>
<Package xmlns="http://soap.sforce.com/2006/04/metadata">
    <types>
        <members>MyAsset</members>
        <name>ContentAsset</name>
    </types>
    <version>48.0</version>
</Package>
```

Wildcard Support in the Manifest File

This metadata type supports the wildcard character `*` (asterisk) in the `package.xml` manifest file. For information about using the manifest file, see Deploying and Retrieving Metadata with the Zip File.

CorsWhitelistOrigin

Represents an origin in the CORS whitelist.

File Suffix and Directory Location

CorsWhitelistOrigin components have the suffix `.corswhitelistorigin` and are stored in the `corswhitelistorigins` folder.

Version

CorsWhitelistOrigin components are available in API version 32.0 and later.

Fields

Field Name	Field Type	Description
`urlPattern`	String	A URL pattern for the origin. The origin URL pattern must include the HTTPS protocol and a domain name, and may include a port. The wildcard character (*) is supported and must be in front of a second-level domain name. For example, `https://*.example.com` adds all subdomains of `example.com` to the whitelist.

Field Name	Field Type	Description
		The origin URL pattern can be an IP address. However, an IP address and a domain that resolve to the same address are not the same origin and you must add them to the CORS whitelist as separate entries.

Declarative Metadata Sample Definition

The following is an example package manifest used to deploy or retrieve the CorsWhitelistOrigin metadata for an organization.

```xml
<?xml version="1.0" encoding="UTF-8"?>
<Package xmlns="http://soap.sforce.com/2006/04/metadata">
    <types>
        <members>*</members>
        <name>CorsWhitelistOrigin</name>
    </types>
    <version>32.0</version>
</Package>
```

The following is an example of a CorsWhitelistOrigin component.

```xml
<?xml version="1.0" encoding="UTF-8"?>
<CorsWhitelistOrigin xmlns="http://soap.sforce.com/2006/04/metadata">
    <developerName>CorsWhitelistEntry1</developerName>
    <urlPattern>https://*.example.com</urlPattern>
</CorsWhitelistOrigin>
```

Usage

CORS (cross-origin resource sharing) is a W3C recommendation that enables Web browsers to request resources from origins other than their own. For example, using CORS, a JavaScript script at `https://www.example.com` could request a resource from `https://www.salesforce.com`.

If a browser that supports CORS makes a request to an origin in the Salesforce CORS whitelist, Salesforce returns the origin in the `Access-Control-Allow-Origin` HTTP header, along with any additional CORS HTTP headers. If the origin is not whitelisted, Salesforce returns HTTP status code 404.

Wildcard Support in the Manifest File

This metadata type supports the wildcard character `*` (asterisk) in the `package.xml` manifest file. For information about using the manifest file, see Deploying and Retrieving Metadata with the Zip File.

CspTrustedSite

Represents a CSP Trusted Site. The Lightning Component framework uses Content Security Policy (CSP) to impose restrictions on content. The main objective is to help prevent cross-site scripting (XSS) and other code injection attacks. To use third-party APIs that make requests to an external (non-Salesforce) server or to use a WebSocket connection, add a CSP Trusted Site. This type extends the Metadata metadata type and inherits its `fullName` field.

Declarative Metadata File Suffix and Directory Location

CspTrustedSite components are stored in the `cspTrustedSites` directory of the corresponding package directory. The file name matches the unique name of the trusted site, and the extension is `.cspTrustedSite`.

Version

CspTrustedSite components are available in API version 39.0 and later.

Fields

Field	Field Type	Description
`context`	CspTrustedSiteContext (enumeration of type string)	Declares the scope of trust for the listed third-party host. • `All` whitelists the host for both Lightning Experience and Lightning Communities experience. • `Communities` whitelists the host for Lightning Communities experience only. • `FieldServiceMobileExtension` whitelists the host for the Field Service Lightning Mobile Extensions only. This value is available in API version 47.0 and later. • (Default) `LEX` whitelists the host for Lightning Experience only. This field is available in API version 44.0 and later.
`description`	string	The description explaining what this trusted site is used for.
`endpointUrl`	string	Required. The URL for the trusted site.
`isActive`	boolean	Required. Indicates if the trusted site is active (`true`) or not (`false`).
`isApplicableToConnectSrc`	boolean	Indicates if Lightning components can load URLs using script interfaces from this site (`true`) or not (`false`). This field has a default value of `false`. Note: At least one CspTrustedSite field beginning with `isApplicable` must be set to `true`. As a result, if all `isApplicable` fields are `false`, these fields all default to `true`. This field is available in API version 48.0 and later.
`isApplicableToFontSrc`	boolean	Indicates if Lightning components can load fonts from this site (`true`) or not (`false`). This field has a default value of `false`. Note: At least one CspTrustedSite field beginning with `isApplicable` must be set to `true`. As a result, if all `isApplicable` fields are `false`, these fields all default to `true`.

Field	Field Type	Description
		This field is available in API version 48.0 and later.
`isApplicableToFrameSrc`	boolean	Indicates if Lightning components can load resources contained in `<iframe>` elements from this site (`true`) or not (`false`). This field has a default value of `false`. Note: At least one CspTrustedSite field beginning with `isApplicable` must be set to `true`. As a result, if all `isApplicable` fields are `false`, these fields all default to `true`. This field is available in API version 48.0 and later.
`isApplicableToImgSrc`	boolean	Indicates if Lightning components can load images from this site (`true`) or not (`false`). This field has a default value of `false`. Note: At least one CspTrustedSite field beginning with `isApplicable` must be set to `true`. As a result, if all `isApplicable` fields are `false`, these fields all default to `true`. This field is available in API version 48.0 and later.
`isApplicableToMediaSrc`	boolean	Indicates if Lightning components can load audio and video from this site (`true`) or not (`false`). This field has a default value of `false`. Note: At least one CspTrustedSite field beginning with `isApplicable` must be set to `true`. As a result, if all `isApplicable` fields are `false`, these fields all default to `true`. This field is available in API version 48.0 and later.
`isApplicableToStyleSrc`	boolean	Indicates if Lightning components can load style sheets from this site (`true`) or not (`false`). This field has a default value of `false`. Note: At least one CspTrustedSite field beginning with `isApplicable` must be set to `true`. As a result, if all `isApplicable` fields are `false`, these fields all default to `true`. This field is available in API version 48.0 and later.
`mobileExtension`	string	Reserved for future use.

Declarative Metadata Sample Definition

A sample XML definition of a trusted site is shown below.

```
<?xml version="1.0" encoding="UTF-8"?>
<CspTrustedSite xmlns="http://soap.sforce.com/2006/04/metadata">
    <description>Used for Lightning component callout to mapping web service</description>

    <endpointUrl>https://www.maptestsite.net/</endpointUrl>
    <isActive>true</isActive>
    <isApplicableToConnectSrc>true</isApplicableToConnectSrc>
    <isApplicableToFontSrc>true</isApplicableToFontSrc>
    <isApplicableToFrameSrc>false</isApplicableToFrameSrc>
    <isApplicableToImgSrc>true</isApplicableToImgSrc>
    <isApplicableToMediaSrc>false</isApplicableToMediaSrc>
    <isApplicableToStyleSrc>true</isApplicableToStyleSrc>
    <context>LEX</context>
</CspTrustedSite>
```

Usage

CSP is a W3C standard that defines rules to control the source of content that can be loaded on a page. All CSP rules work at the page level, and apply to all components and libraries. By default, the framework's headers allow content to be loaded only from secure (HTTPS) URLs and forbid XHR requests from JavaScript.

When you define a CSP Trusted Site, you can add the site's URL to the list of allowed sites for the following directives in the CSP header.

- `connect-src`
- `frame-src`
- `img-src`
- `style-src`
- `font-src`
- `media-src`

This change to the CSP header directives allows Lightning components to load resources, such as images, styles, and fonts, from the site. It also allows client-side code to make requests to the site.

Wildcard Support in the Manifest File

This metadata type supports the wildcard character `*` (asterisk) in the `package.xml` manifest file. For information about using the manifest file, see Deploying and Retrieving Metadata with the Zip File.

CustomApplication

CustomApplication represents a custom or standard application. In API version 29.0 and earlier, CustomApplication represents only a custom application. An application is a list of tab references, with a description and a logo. This type extends the Metadata metadata type and inherits its `fullName` field.

File Suffix and Directory Location

Custom and standard applications have the suffix `.app` and are stored in the `applications` folder.

Note: Retrieving a component of this metadata type in a project makes the component appear in any Profile and PermissionSet components that are retrieved in the same package.

Version

Custom applications are available in API version 10.0 and later. Standard applications are available in API version 30.0 and later.

Fields

Field Name	Field Type	Description
`actionOverrides`	AppActionOverride[]	Represents an action override for an application. Use it to create, update, edit, or delete action overrides. This field is available for Lightning Experience in API version 38.0 and later.
`brand`	AppBrand	The color scheme and logo used for the app. This field is available for Lightning Experience in API version 38.0 and later.
`consoleConfig`	ServiceCloudConsoleConfig	Represents configuration settings for a Salesforce console app. This field is available in API version 42.0 and later.
`defaultLandingTab`	string	The `fullName` of a standard tab or custom tab that opens when this application is selected.
`description`	string	The optional description text of the application.
`formFactors`	FormFactor (enumeration of type string)	Indicates the form factors for which the app is visible for Lightning Experience. Valid values are: • `Null` (no value)—For a desktop using Salesforce Classic • `Small`—For a mobile device using the Salesforce mobile app • `Medium`—Reserved for future use • `Large`—For a desktop using Lightning Experience This field is available in API version 38.0 and later. Note: As of version 38.0, `formFactors` is set to `Large` for existing Salesforce Classic apps, except for Salesforce Classic consoles. Salesforce Classic apps installed from packages created before version 38.0 also have `formFactors`

Field Name	Field Type	Description
		set to `Large`. For Salesforce Classic apps in packages created with 38.0 or later, you must set `formFactors` to `Large` for Salesforce Classic apps to appear in the Lightning Experience desktop. As of API version 47.0, the `Small` value is supported for Lightning apps. The `formFactors` field can be set to `Small` or `Large` for Lightning apps, and it can be set to `Null` or `Large` for Salesforce Classic apps.
`isNavAutoTempTabsDisabled`	boolean	Indicates whether the navigation automatically creates temporary tabs settings. Applies only to Lightning apps with standard navigation. Available in API version 43.0 and later.
`isNavPersonalizationDisabled`	boolean	Indicates whether navigation personalization is disabled. Applies only to Lightning apps. Available in API version 43.0 and later.
`isServiceCloudConsole`	boolean	Indicates if the application is a Salesforce Classic console app. For Lightning Experience console apps, this field is `null` and the `navType` field is set to `Console`.
`label`	string	The name of the application.
`logo`	string	The optional reference to the image document for a Salesforce app or Salesforce console app.
`navType`	NavType (enumeration of type string)	Not updateable. Indicates the type of navigation the app uses. The value `Standard` is for a Lightning app with standard navigation. The value `Console` is for a Lightning app with console navigation. This field is available in API version 38.0 and later.
`preferences`	AppPreferences	Represents the preferences for a Salesforce Classic console app. All of the AppPreferences fields are required. This field is available in API version 42.0 and later.
`profileActionOverrides`	AppProfileActionOverride[]	A list of the Lightning Experience record page ProfileActionOverrides that are assigned to this custom app. When a user invokes the custom app, a matching ProfileActionOverride assignment takes precedence over existing overrides for the record page specified in ActionOverride. This lets you override a record page for the custom app by record type and profile. In API version 45.0 and later, you can override a home page for the custom app by profile.

Field Name	Field Type	Description
setupExperience	string	The type of Setup experience associated with the app. Valid values are: • all—Represents the full Setup tree. • essentials—Represents the Essentials Setup tree, which contains a subset of Setup items configured for Essentials edition. • service—Represents the Service Setup tree, which contains a subset of Setup items configured for Service Console. A null value is equivalent to all. Previous valid values AllSetup, ServiceSetup, and EssentialsSetup have been deprecated. This field is available in API version 39.0 and later.
subscriberTabs	string[]	Represents the list of tabs appended by a subscriber to a Lightning app installed from a managed package. Records in a subscriber tab always open as primary tabs. This field is available in API version 41.0 and later.
tabs	string[]	The list of tabs included in this application. In API version 12.0, the fullName for built-in tabs like Home, Account, and Reports, is the name of the tab (Home, for example). In API version 13.0 and later, built-in tabs are prefixed with standard-. For example, to reference the Account tab you would use standard-Account. In API version 42.0, this field was renamed from tab to tabs.
uiType	UiType (enumeration of type string)	Not updateable. Identifies the type of custom app. The value is: • Aloha for Salesforce Classic • Lightning for Lightning Experience This field is available in API version 38.0 and later.
utilityBar	string	The developer name of the utility bar associated with this app. Note: We recommend assigning a utility bar to only one Lightning App, because utility bars are shared. Sharing means that if you change the utility bar in one app, it automatically changes in all apps that it’s part of. This field is available in API version 38.0 and later.

Field Name	Field Type	Description
`workspaceConfig`	AppWorkspaceConfig	Represents how records open in a Salesforce console app. Required if `isServiceCloudConsole` is `true`. In API version 42.0, this field was renamed to `workspaceConfig` from `workspaceMappings`.

AppActionOverride

Represents an action override for an application. Use it to create, update, edit, or delete action overrides. AppActionOverride inherits from ActionOverride and extends it by one field, `pageOrSobjectType`. Available for Lightning Experience in API version 38.0 and later.

Field Name	Field Type	Description
`actionName`	string	The only valid value is `view` for API version 43.0 and earlier. The value `tab` is supported for API version 44.0 and later.
`comment`	string	Any comments you want associated with the override.
`content`	string	Set this field if `type` is set to `flexipage`. It refers to the name of the page to use as the override. To reference installed components, use the format of ***`Component_namespace__Component_name`***.
`formFactor`	FormFactor (enumeration of type string)	The size of the page being overridden. If the `type` field is set to `flexipage`, set this field to `Large` to override the View action with a Lightning page in Lightning Experience. The `Large` value represents the Lightning Experience desktop environment and is valid only for the `flexipage` and `lightningcomponent` types. The `Small` value represents the Salesforce app on a phone or tablet. The `Medium` value is reserved for future use. The `null` value (which is the same as specifying no value) represents Salesforce Classic. This field is available in API version 37.0 and later and is part of the feature for creating and editing record pages in Lightning Experience. Note: Lightning component overrides return different `FormFactor` values depending on the API version used. • In API version 41.0 and earlier, Lightning component overrides return only the `null` value (no value), representing the Salesforce Classic environment. • In API version 42.0, if you specify different Lightning component overrides for Lightning Experience and mobile, one component is selected randomly for both overrides and its `FormFactor` value is returned. If there's a conflict between Lightning components, and a Visualforce page override is also specified for Salesforce Classic, the Visualforce page takes precedence.

Field Name	Field Type	Description
		- In API version 43.0 and later, a Lightning component override for Lightning Experience returns the `Large` value and a Lightning component override for mobile returns the `Small` value, as expected.
`pageOrSobjectType`	string	The name of the sObject type being overridden. Valid values are `standard` and `custom`. This value must be `standard-home` when actionName is `tab`.
`skipRecordTypeSelect`	boolean	Set this field to `true` if you prefer that any new records created by this action override aren't forwarded to the record type selection page. This field is only valid if the `actionName` is a "create" type (like `new`), and `type` is set to `visualforce`.
`type`	ActionOverrideType (enumeration of type string)	Required. Represents the type of action override. The valid values are `Flexipage` and `Default`. A `Flexipage` AppActionOverride set to App Default can't be deleted via Metadata API. Instead, remove the override using the page assignment wizard in the Lightning App Builder UI.

AppBrand

The color scheme and logo used for the app. Available for Lightning apps in API version 38.0 and later.

Field Name	Field Type	Description
`footerColor`	string	Optional. Determines the footer color in the app. Specify the color with a hexadecimal code, such as #0000FF for blue.
`headerColor`	string	Optional. Determines the header color in the app. Specify the color with a hexadecimal code, such as #0000FF for blue.
`logo`	string	The optional reference to the image document for the application.
`logoVersion`	int	An optional version number for the logo.
`shouldOverrideOrgTheme`	boolean	Indicates whether to override the global theme for the org. When `true`, the color scheme and logo that the user has set are used. When `false`, the global theme for the org is used, even if the user has set a color scheme and logo.

AppComponentList

Represents custom console components (Visualforce pages) assigned to a Salesforce console app. In API version 42.0, this type was renamed from CustomApplicationComponents to AppComponentList.

Field Name	Field Type	Description
`alignment`	string	Required. Determines how custom console components are aligned in the footer of a Salesforce console app.
`components`	string[]	The name of a custom console component assigned to a Salesforce console app. In API version 42.0, this field was renamed from `customApplicationComponent` to `components`.

AppPreferences

Represents the preferences for a Salesforce Classic console app. All of the AppPreferences fields are required. Available in API version 42.0 and later.

Field Name	Field Type	Description
`enableCustomizeMyTabs`	boolean	Indicates if a Salesforce Classic console app has Customize My Tabs enabled. If enabled, users can hide, display, and organize items in the navigation tab.
`enableKeyboardShortcuts`	boolean	Indicates if a Salesforce Classic console app has keyboard shortcuts enabled. Shortcuts let users perform actions by pressing a combination of keys instead of having to use a mouse. After keyboard shortcuts are enabled, several default shortcuts are available for customization. Before you can create custom shortcuts, a developer must define the shortcut's action with the `addEventListener()` method in the Salesforce Console Integration Toolkit. You can't create keyboard shortcuts for actions performed outside of the console. This field is required if `isServiceCloudConsole` is `true`.
`enableListViewHover`	boolean	Indicates if a Salesforce Classic console app has list view hovers enabled. If set to `true`, summary information is displayed about a record in a responsive list when the user hovers over a record name. For cases, hover over the subject field.
`enableListViewReskin`	boolean	Indicates if Salesforce Classic console apps use responsive list views instead of Salesforce Classic lists views.
`enableMultiMonitorComponents`	boolean	Indicates if a Salesforce Classic console app has multi-monitor components enabled, which lets users move portions of a console from their browsers to locations on their screens. This field is required if `isServiceCloudConsole` is `true`.
`enablePinTabs`	boolean	Indicates if a Salesforce Classic console app has pinned tabs enabled, which lets users pin primary tabs to the tab bar for quick access.
`enableTabHover`	boolean	Indicates if a Salesforce Classic console app has tab hover enabled. If enabled, summary information is displayed about a record in an overlay when the user hovers over a tab.

Field Name	Field Type	Description
enableTabLimits	boolean	Indicates whether limits are enabled on the number of primary tabs and subtabs that can be opened in a Salesforce Classic console session. When true, values for tabLimitConfig are required
saveUserSessions	boolean	Indicates if a Salesforce Classic console app saves user sessions automatically. If enabled, when console users close their browsers or log out of Salesforce, any previously open tabs display when users log in again. Required if isServiceCloudConsole is true.

AppProfileActionOverride

Represents a ProfileActionOverride for a custom app. This type inherits from ProfileActionOverride on page 746 and extends it by one field, profile. Available for Lightning Experience in API version 39.0 and later. In API version 45.0 and later, you can override a home page for the custom app by profile.

Field Name	Field Type	Description
actionName	string	Required. The name of the action. The only valid values are Tab and View. If pageOrSobjectType is record-home, this field must be View. The View action is supported only when ProfileActionOverride is being specified as part of a CustomApplication. In API version 45.0 and later, this action is supported only when ProfileActionOverride is being specified as part of a CustomApplication, pageOrSobjectType is standard-home, and this field is Tab.
content	string	Read-only. Represents the name of the Lightning page being used as the override.
formFactor	FormFactor (enumeration of type string)	Required. The size of the page being overridden. The Large value represents the Lightning Experience desktop environment.
pageOrSobjectType	string	Required. The name of the page being overridden. The only valid values are record-home and standard-home. If the actionName is Tab, this field must be standard-home
profile	string	The profile associated with the ProfileActionOverride.
recordType	string	The record type associated with the override. If pageOrSobjectType is standard-home, this field must be null. This field is required when actionName is set to View.
type	ActionOverrideType (enumeration of type string)	Required. Read-only. The type of action override. The only valid value is flexipage.

AppWorkspaceConfig

Represents how records open in a Salesforce console app. Required if `isServiceCloudConsole` is `true`. Available for Salesforce Classic console apps in API version 25.0 and later. Available for Lightning console apps in API version 41.0 and later. In API version 42.0, this type was renamed from WorkspaceMappings to AppWorkspaceConfig.

Field Name	Field Type	Description
`mappings`	WorkspaceMapping[]	Represents how records for a specific tab open in a Salesforce console app. Required for each tab specified in the CustomApplication. In API version 42.0, this field was renamed from `workspaceMapping` to `mappings`.

WorkspaceMapping

Represents how records for a specific tab open in a Salesforce console app. Required for each tab specified in the CustomApplication. Available in API version 25.0 and later for Salesforce Classic console apps. Available in API version 41.0 and later for Lightning console apps.

Field Name	Field Type	Description
`fieldName`	string	The name of the field that specifies the primary tab in which to display `tab` as a subtab. If not specified, `tab` opens as a primary tab.
`tab`	string	Required. Name of the tab.

CustomShortcut

Represents custom keyboard shortcuts assigned to a Salesforce console app in Salesforce Classic. Before you can create custom shortcuts, a developer must define the shortcut's action with the `addEventListener()` method in the Salesforce Console Integration Toolkit. You can't create keyboard shortcuts for actions performed outside of the console. Available in API version 28.0 and later.

Field Name	Field Type	Description
`action`	string	Required. The action performed in the console when a user presses the keyboard shortcut.
`active`	boolean	Required. Indicates whether the keyboard shortcut is active (`true`) or not (`false`).
`keyCommand`	string	Required. The combination of keys a user presses to trigger the keyboard shortcut. Keyboard shortcuts aren't case-sensitive, but they display as uppercase on setup pages in the Salesforce user interface so that they're easier to read. Each key command can include up to four modifier keys followed by one non-modifier key. Modifier and non-modifier keys are separated by the + key. Modifier keys can occur in any order, but you must place non-modifier keys at the end of the key command sequence. For example, `SHIFT+CTRL+ALT+META +A`.

Field Name	Field Type	Description
		Valid modifier keys are: • `SHIFT` • `CTRL` • `ALT` • `META` (represents the COMMAND key on Macs) Valid non-modifier keys are letters A through Z and numbers 0 through 9. Other valid keys are: • `TAB` • `ENTER` • `PAUSE/BREAK` • `CAPS LOCK` • `ESC` • `SPACE` • `PAGE UP` • `PAGE DOWN` • `END` • `HOME` • `LEFT ARROW` • `UP ARROW` • `RIGHT ARROW` • `DOWN ARROW` • `PRINT SCREEN` • `INSERT` • `DELETE` • `RIGHT WINDOW` • `NUMPAD 0` • `NUMPAD 1` • `NUMPAD 2` • `NUMPAD 3` • `NUMPAD 4` • `NUMPAD 5` • `NUMPAD 6` • `NUMPAD 7` • `NUMPAD 8` • `NUMPAD 9` • `MULTIPLY` • `ADD` • `SUBTRACT`

Field Name	Field Type	Description
		• DECIMAL POINT • DIVIDE • F1 • F2 • F3 • F4 • F5 • F6 • F7 • F8 • F9 • F10 • F11 • F12 • NUM LOCK • SCROLL LOCK • ; • = • , • — • . • / • ` • [•] • \ • '
description	string	The optional description text for the keyboard shortcut.
eventName	string	Required. Code available to developers who want to add custom shortcut functions to the console via the Salesforce Console Integration Toolkit.

DefaultShortcut

Represents default keyboard shortcuts assigned to a Salesforce console app. Once you enable keyboard shortcuts for a console, several default shortcuts are available for customization. These include opening and closing tabs, moving between tabs, and saving records. Available in API version 28.0 and later.

Field Name	Field Type	Description
action	string	Required. The action performed in the console when a user presses the keyboard shortcut. Valid values are: • FOCUS_CONSOLE • FOCUS_NAVIGATOR_TAB • FOCUS_DETAIL_VIEW • FOCUS_PRIMARY_TAB_PANEL • FOCUS_SUBTAB_PANEL • FOCUS_LIST_VIEW • FOCUS_FIRST_LIST_VIEW • FOCUS_SEARCH_INPUT • MOVE_LEFT • MOVE_RIGHT • UP_ARROW • DOWN_ARROW • OPEN_TAB_SCROLLER_MENU • OPEN_TAB • CLOSE_TAB • ENTER • EDIT • SAVE For a list and description of the default keyboard shortcuts, see “Default Keyboard Shortcuts for a Salesforce Console in Salesforce Classic” in Salesforce Help.
active	boolean	Required. Indicates whether the keyboard shortcut is active (true) or not (false).
keyCommand	string	Required. The combination of keys a user presses to trigger the keyboard shortcut. Keyboard shortcuts aren’t case-sensitive, but they display as uppercase on setup pages in the Salesforce user interface so that they’re easier to read. Each key command can include up to four modifier keys followed by one non-modifier key. Modifier and non-modifier keys are separated by the + key. Modifier keys can occur in any order, but you must place non-modifier keys at the end of the key command sequence. For example, SHIFT+CTRL+ALT+META +A. Valid modifier keys are: • SHIFT • CTRL • ALT • META (represents the COMMAND key on Macs)

Field Name	Field Type	Description
		Valid non-modifier keys are letters A through Z and numbers 0 through 9. Other valid keys are: • `TAB` • `ENTER` • `PAUSE/BREAK` • `CAPS LOCK` • `ESC` • `SPACE` • `PAGE UP` • `PAGE DOWN` • `END` • `HOME` • `LEFT ARROW` • `UP ARROW` • `RIGHT ARROW` • `DOWN ARROW` • `PRINT SCREEN` • `INSERT` • `DELETE` • `RIGHT WINDOW` • `NUMPAD 0` • `NUMPAD 1` • `NUMPAD 2` • `NUMPAD 3` • `NUMPAD 4` • `NUMPAD 5` • `NUMPAD 6` • `NUMPAD 7` • `NUMPAD 8` • `NUMPAD 9` • `MULTIPLY` • `ADD` • `SUBTRACT` • `DECIMAL POINT` • `DIVIDE` • `F1` • `F2` • `F3`

Field Name	Field Type	Description
		• F4 • F5 • F6 • F7 • F8 • F9 • F10 • F11 • F12 • NUM LOCK • SCROLL LOCK • ; • = • , • - • . • / • ` • [•] • \ • '

KeyboardShortcuts

Represents keyboard shortcuts assigned to a Salesforce console app. Required if `isServiceCloudConsole` is `true`. Available in API version 28.0 and later.

Field Name	Field Type	Description
`customShortcuts`	CustomShortcut[]	Represents custom keyboard shortcuts assigned to a Salesforce console app in Salesforce Classic. Before you can create custom shortcuts, a developer must define the shortcut's action with the `addEventListener()` method in the Salesforce Console Integration Toolkit. You can't create keyboard shortcuts for actions performed outside of the console. In API version 42.0, this field was renamed from `customShortcut` to `customShortcuts`.
`defaultShortcuts`	DefaultShortcut[]	Represents default keyboard shortcuts assigned to a Salesforce console app. Once you enable keyboard shortcuts for a console, several default

Field Name	Field Type	Description
		shortcuts are available for customization. These include opening and closing tabs, moving between tabs, and saving records. For a list and description of the default keyboard shortcuts, see "Default Keyboard Shortcuts for a Salesforce Console in Salesforce Classic" in Salesforce Help. In API version 42.0, this field was renamed from `defaultShortcut` to `defaultShortcuts`.

ListPlacement

Represents how lists display in a Salesforce console app. Required if `isServiceCloudConsole` is `true`. Available in API version 25.0 and later.

Field Name	Field Type	Description
`height`	int	Height of the list in pixels or percentage. Required if `location` is top.
`location`	string	Required. Location of the list on the screen. Valid values are: • full • top • left
`units`	string	Required. Represents if `height` or `width` is in pixels or percentage.
`width`	int	Width of the list in pixels or percentage. Required if `location` is left.

LiveAgentConfig

Represents your organization's settings for using Chat in the Salesforce Console.

Field Name	Field Type	Description
`enableLiveChat`	boolean	Specifies whether Chat is enabled in your organization (`true`) or not (`false`).
`openNewAccountSubtab`	boolean	Specifies whether to open a new Account subtab in a Salesforce console app automatically (`true`) or not (`false`) when an agent accepts a chat.
`openNewCaseSubtab`	boolean	Specifies whether to open a new Case subtab in a Salesforce console app automatically (`true`) or not (`false`) when an agent accepts a chat.
`openNewContactSubtab`	boolean	Specifies whether to open a new Contact subtab in a Salesforce console app automatically (`true`) or not (`false`) when an agent accepts a chat.

Field Name	Field Type	Description
openNewLeadSubtab	boolean	Specifies whether to open a new Lead subtab in a Salesforce console app automatically (true) or not (false) when an agent accepts a chat.
openNewVFPageSubtab	boolean	Specifies whether to open a new Visualforce page as a subtab in a Salesforce console app automatically (true) or not (false) when an agent accepts a chat.
pageNamesToOpen	string [array of strings]	Specifies the Visualforce pages to open in subtabs when an agent accepts a chat in a Salesforce console app. This field is available in API version 42.0 and later.
showKnowledgeArticles	boolean	Specifies whether to display the Knowledge component while using Chat in a Salesforce console app (true) or not (false).

PushNotification

Represents a set of push notifications, which are visual indicators on lists and detail pages that show when a record or field has changed during a user's session. Available for use if isServiceCloudConsole is true. Available in API version 28.0 and later.

Field Name	Field Type	Description
fieldNames	string]	The name of the field, or fields, that trigger push notifications for the selected object.
objectName	string	Required. Name of the object that triggers push notifications.

ServiceCloudConsoleConfig

Represents configuration settings for a Salesforce console app. Available in API version 42.0 and later.

Field Name	Field Type	Description
componentList	AppComponentList	Represents custom console components (Visualforce pages) assigned to a Salesforce console app.
detailPageRefreshMethod	string	Determines how detail pages refresh in a Salesforce console app. Required if isServiceCloudConsole is true. The valid values are: • none • autoRefresh • flag
footerColor	string	Determines the footer color in a Salesforce console app. Specify the color with a hexadecimal code, such as #0000FF for blue.
headerColor	string	Determines the header color in a Salesforce console app. Specify the color with a hexadecimal code, such as #0000FF for blue.

Field Name	Field Type	Description
keyboardShortcuts	KeyboardShortcuts	Represents the keyboard shortcuts for a Salesforce console app. Keyboard shortcuts let users perform actions by pressing a combination of keys instead of having to use a mouse.
listPlacement	ListPlacement	Represents how lists display in a Salesforce console app. Required if isServiceCloudConsole is true.
listRefreshMethod	string	Determines how lists refresh in a Salesforce console app. Required if isServiceCloudConsole is true. The valid values are: • none • refreshList • refreshListRows
liveAgentConfig	LiveAgentConfig	Represents the configurations for using Chat in the Salesforce Console.
primaryTabColor	string	Determines the primary tab color in a Salesforce console app. Specify the color with a hexadecimal code, such as #0000FF for blue.
pushNotifications	PushNotification[]	Represents push notifications for a Salesforce console app. Push notifications are visual indicators on lists and detail pages that show when a record or field has changed during a user's session. For example, assume that two support agents are working on the same case. If one agent changes the Priority, a push notification displays to the other agent so the agent notices the change and doesn't duplicate the effort.
tabLimitConfig	TabLimitConfig	Represents the maximum number of primary tabs and subtabs allowed in one Salesforce console session. Required if enableTabLimits is true.
whiteListedDomains	string[]	Any external domains that users can access from within a Salesforce console app. For example, www.yourdomain.com.

TabLimitConfig

Represents the maximum number of primary tabs and subtabs allowed in one Salesforce console session. Required if enableTabLimits is true. Available in API version 36.0 and later.

Field Name	Field Type	Description
maxNumberOfPrimaryTabs	string	The maximum number of primary tabs allowed in one console session. Valid values are: • 5 • 10 • 20 • 30

Field Name	Field Type	Description
maxNumberOfSubTabs	string	The maximum number of subtabs allowed in one console session. Valid values are: • 5 • 10 • 15

Retrieving Apps

To retrieve apps in your organization, use the CustomApplication type name in the `package.xml` manifest file. You can either retrieve all apps or specify which apps to retrieve in the types section of `package.xml`.

To retrieve all apps in your organization—custom and standard apps, specify the wildcard character (`*`), as follows.

```
<types>
    <members>*</members>
    <name>CustomApplication</name>
</types>
```

Note: In API version 29.0 and earlier, use of the wildcard returns only all custom applications but not standard applications.

To retrieve a custom app, specify the app name.

```
<types>
    <members>MyCustomApp</members>
    <name>CustomApplication</name>
</types>
```

To retrieve a standard app, add the `standard__` prefix to the app name. For example, to retrieve the Chatter standard app, specify `standard__Chatter`.

```
<types>
    <members>standard__Chatter</members>
    <name>CustomApplication</name>
</types>
```

To retrieve an app that is part of an installed package, add the package namespace prefix followed by two underscores and the app name. For example, if the package namespace is `myInstalledPackageNS` and the app name is `PackageApp`, specify `myInstalledPackageNS__PackageApp`, as follows.

```
<types>
    <members>myInstalledPackageNS__PackageApp</members>
    <name>CustomApplication</name>
</types>
```

Declarative Metadata Sample Definition

Here's the definition of a custom Lightning Experience app:

```
<?xml version="1.0" encoding="UTF-8"?>
<CustomApplication xmlns="http://soap.sforce.com/2006/04/metadata">
```

```
    <actionOverrides>
        <actionName>View</actionName>
        <comment>Action override created by Lightning App Builder during
activation.</comment>
        <content>Custom_Mobile_Oppty_Page</content>
        <formFactor>Small</formFactor>
        <skipRecordTypeSelect>false</skipRecordTypeSelect>
        <type>Flexipage</type>
        <pageOrSobjectType>Opportunity</pageOrSobjectType>
    </actionOverrides>
    <actionOverrides>
        <actionName>View</actionName>
        <comment>Action override created by Lightning App Builder during
activation.</comment>
        <content>Custom_Mobile_Oppty_Page</content>
        <formFactor>Large</formFactor>
        <skipRecordTypeSelect>false</skipRecordTypeSelect>
        <type>Flexipage</type>
        <pageOrSobjectType>Opportunity</pageOrSobjectType>
    </actionOverrides>
    <brand>
        <headerColor>#EE1518</headerColor>
        <shouldOverrideOrgTheme>true</shouldOverrideOrgTheme>
    </brand>
    <description>Manage inventory and deliveries for our warehouses.</description>
    <formFactors>Small</formFactors>
    <formFactors>Large</formFactors>
    <isNavAutoTempTabsDisabled>false</isNavAutoTempTabsDisabled>
    <isNavPersonalizationDisabled>false</isNavPersonalizationDisabled>
    <label>Warehouse Lightning</label>
    <navType>Standard</navType>
    <profileActionOverrides>
        <actionName>View</actionName>
        <content>Warehouse_test_page</content>
        <formFactor>Large</formFactor>
        <pageOrSobjectType>Warehouse__c</pageOrSobjectType>
        <type>Flexipage</type>
        <profile>Admin</profile>
    </profileActionOverrides>
    <profileActionOverrides>
        <actionName>View</actionName>
        <content>Warehouse_test_page</content>
        <formFactor>Small</formFactor>
        <pageOrSobjectType>Warehouse__c</pageOrSobjectType>
        <type>Flexipage</type>
        <profile>Admin</profile>
    </profileActionOverrides>
    <setupExperience>all</setupExperience>
    <tabs>standard-Feed</tabs>
    <tabs>standard-File</tabs>
    <tabs>standard-Account</tabs>
    <tabs>standard-Case</tabs>
    <tabs>Merchandise__c</tabs>
    <tabs>Invoice__c</tabs>
    <tabs>Warehouse__c</tabs>
    <tabs>Delivery__c</tabs>
    <tabs>standard-report</tabs>
    <tabs>standard-Dashboard</tabs>
    <uiType>Lightning</uiType>
</CustomApplication>
```

The following is a definition of a standard app (Chatter):

```
<?xml version="1.0" encoding="UTF-8"?>
<CustomApplication xmlns="http://soap.sforce.com/2006/04/metadata">
    <defaultLandingTab>standard-home</defaultLandingTab>
    <label>Collaboration</label>
    <tabs>standard-Chatter</tabs>
    <tabs>standard-UserProfile</tabs>
    <tabs>standard-OtherUserProfile</tabs>
    <tabs>standard-CollaborationGroup</tabs>
    <tabs>standard-File</tabs>
</CustomApplication>
```

Declarative Metadata Sample Definition—Salesforce Console

The following is the definition of a custom app where `isServiceCloudConsole` is `true`:

```
<?xml version="1.0" encoding="UTF-8"?>
<CustomApplication xmlns="http://soap.sforce.com/2006/04/metadata">
    <consoleConfig>
        <componentList>
            <alignment>left</alignment>
            <components>MyComponent</components>
        </componentList>
        <detailPageRefreshMethod>autoRefresh</detailPageRefreshMethod>
        <keyboardShortcuts>
            <customShortcuts>
                <action>MyCustomShortcutAction</action>
                <active>true</active>
                <keyCommand>X</keyCommand>
                <description>Custom Shortcut example</description>
                <eventName>myCustomShortcutExample</eventName>
            </customShortcuts>
            <defaultShortcuts>
                <action>FOCUS_CONSOLE</action>
                <active>true</active>
                <keyCommand>ESC</keyCommand>
            </defaultShortcuts>
            <defaultShortcuts>
                <action>FOCUS_NAVIGATOR_TAB</action>
                <active>true</active>
                <keyCommand>V</keyCommand>
            </defaultShortcuts>
            <defaultShortcuts>
                <action>FOCUS_DETAIL_VIEW</action>
                <active>true</active>
                <keyCommand>SHIFT+S</keyCommand>
            </defaultShortcuts>
            <defaultShortcuts>
                <action>FOCUS_PRIMARY_TAB_PANEL</action>
                <active>true</active>
                <keyCommand>P</keyCommand>
            </defaultShortcuts>
            <defaultShortcuts>
                <action>FOCUS_SUBTAB_PANEL</action>
                <active>true</active>
                <keyCommand>S</keyCommand>
            </defaultShortcuts>
            <defaultShortcuts>
                <action>FOCUS_LIST_VIEW</action>
                <active>true</active>
                <keyCommand>N</keyCommand>
            </defaultShortcuts>
            <defaultShortcuts>
                <action>FOCUS_FIRST_LIST_VIEW</action>
                <active>true</active>
                <keyCommand>SHIFT+F</keyCommand>
            </defaultShortcuts>
            <defaultShortcuts>
                <action>FOCUS_SEARCH_INPUT</action>
                <active>true</active>
                <keyCommand>R</keyCommand>
            </defaultShortcuts>
            <defaultShortcuts>
                <action>MOVE_LEFT</action>
                <active>true</active>
                <keyCommand>LEFT ARROW</keyCommand>
            </defaultShortcuts>
            <defaultShortcuts>
                <action>MOVE_RIGHT</action>
                <active>true</active>
                <keyCommand>RIGHT ARROW</keyCommand>
            </defaultShortcuts>
            <defaultShortcuts>
                <action>UP_ARROW</action>
                <active>true</active>
                <keyCommand>UP ARROW</keyCommand>
            </defaultShortcuts>
            <defaultShortcuts>
                <action>DOWN_ARROW</action>
                <active>true</active>
                <keyCommand>DOWN ARROW</keyCommand>
            </defaultShortcuts>
            <defaultShortcuts>
                <action>OPEN_TAB_SCROLLER_MENU</action>
                <active>true</active>
                <keyCommand>D</keyCommand>
            </defaultShortcuts>
            <defaultShortcuts>
                <action>OPEN_TAB</action>
                <active>true</active>
                <keyCommand>T</keyCommand>
            </defaultShortcuts>
            <defaultShortcuts>
                <action>CLOSE_TAB</action>
                <active>true</active>
                <keyCommand>C</keyCommand>
            </defaultShortcuts>
            <defaultShortcuts>
                <action>ENTER</action>
                <active>true</active>
                <keyCommand>ENTER</keyCommand>
            </defaultShortcuts>
            <defaultShortcuts>
                <action>EDIT</action>
                <active>true</active>
                <keyCommand>E</keyCommand>
            </defaultShortcuts>
            <defaultShortcuts>
                <action>SAVE</action>
                <active>true</active>
                <keyCommand>CTRL+S</keyCommand>
            </defaultShortcuts>
        </keyboardShortcuts>
        <listPlacement>
            <location>left</location>
            <units>percent</units>
            <width>20</width>
        </listPlacement>
        <listRefreshMethod>refreshList</listRefreshMethod>
        <pushNotifications>
            <fieldNames>CreatedBy</fieldNames>
            <objectName>Campaign</objectName>
        </pushNotifications>
        <pushNotifications>
            <fieldNames>CustomField1__c</fieldNames>
            <objectName>CustomObject1__c</objectName>
        </pushNotifications>
    </consoleConfig>
    <defaultLandingTab>standard-home</defaultLandingTab>
    <isServiceCloudConsole>true</isServiceCloudConsole>
    <label>MyConsole</label>
    <preferences>
        <enableCustomizeMyTabs>false</enableCustomizeMyTabs>
        <enableKeyboardShortcuts>true</enableKeyboardShortcuts>
        <enableListViewHover>true</enableListViewHover>
        <enableListViewReskin>true</enableListViewReskin>
        <enableMultiMonitorComponents>true</enableMultiMonitorComponents>
        <enablePinTabs>true</enablePinTabs>
        <enableTabHover>false</enableTabHover>
        <enableTabLimits>false</enableTabLimits>
        <saveUserSessions>false</saveUserSessions>
    </preferences>
    <tabs>standard-Case</tabs>
    <tabs>standard-Account</tabs>
    <tabs>standard-Contact</tabs>
    <tabs>standard-Contract</tabs>
    <workspaceConfig>
        <mappings>
            <tab>standard-Case</tab>
        </mappings>
        <mappings>
            <fieldName>ParentId</fieldName>
            <tab>standard-Account</tab>
        </mappings>
        <mappings>
            <fieldName>AccountId</fieldName>
            <tab>standard-Contact</tab>
        </mappings>
        <mappings>
            <tab>standard-Contract</tab>
        </mappings>
    </workspaceConfig>
</CustomApplication>
```

Wildcard Support in the Manifest File

This metadata type supports the wildcard character `*` (asterisk) in the `package.xml` manifest file. For information about using the manifest file, see Deploying and Retrieving Metadata with the Zip File.

SEE ALSO:

CustomTab

CustomApplicationComponent

Represents a custom console component (Visualforce page) assigned to a CustomApplication that is marked as a Salesforce console. Custom console components extend the capabilities of Salesforce console apps. See “Customize a Console with Custom Components in Salesforce Classic” in the Salesforce online help.

File Suffix and Directory Location

Custom application components have the suffix `.customApplicationComponent` and are stored in the `customApplicationComponents` folder.

Version

Custom applications are available in API version 25.0 and later.

Fields

Field Name	Field Type	Description
buttonIconUrl	string	The address of a page that hosts an icon for the button.
buttonStyle	string	The inline style used to define how the button looks.
buttonText	string	The label on the button used to launch the custom console component.
buttonWidth	int	The pixel width of the button as it should display in the Salesforce console.
height	int	The pixel height of the window used to display the custom console component.
isHeightFixed	boolean	Required. Indicates whether users can change the custom console component height (false) or not (true).
isHidden	boolean	Required. Indicates whether the custom console component is hidden from users (true) or not (false).
isWidthFixed	boolean	Required. Indicates whether users can change the component width (false) or not (true).
visualforcePage	string	Required. Name of the Visualforce page that represents the custom console component.
width	int	The pixel width of the window used to display the custom console component.

Declarative Metadata Sample Definition

The following is the definition of a custom application component:

```
<?xml version="1.0" encoding="UTF-8"?>
<CustomApplicationComponent xmlns="http://soap.sforce.com/2006/04/metadata">
    <buttonIconUrl>http://www.salesforce.com</buttonIconUrl>
    <buttonStyle>buttonStyleCSS</buttonStyle>
    <buttonText>buttonText</buttonText>
    <buttonWidth>200</buttonWidth>
    <height>200</height>
    <isHeightFixed>false</isHeightFixed>
    <isHidden>false</isHidden>
    <isWidthFixed>false</isWidthFixed>
    <visualforcePage>MyVisualforcePage</visualforcePage>
    <width>50</width>
</CustomApplicationComponent>
```

Wildcard Support in the Manifest File

This metadata type supports the wildcard character * (asterisk) in the package.xml manifest file. For information about using the manifest file, see Deploying and Retrieving Metadata with the Zip File.

CustomFeedFilter

Represents a custom feed filter that limits the feed view to feeds from the Cases object. The custom feed filter shows only feed items that satisfy the criteria specified in the CustomFeedFilter definition.This type extends the Metadata metadata type and inherits its `fullName` field.

File Suffix and Directory Location

CustomFeedFilter components have the suffix `.feedFilter` and are stored in the `feedFilters` folder.

Version

CustomFeedFilter components are available in API version 35.0 and later.

Fields

Field Name	Field Type	Description
`criteria`	FeedFilterCriterion []	The criterion that defines which feed items are shown when the filter is applied. The feed filter displays all feed items that satisfy the criteria.
`description`	string	The description of the custom feed filter. For example, specify what feed items that filter shows.
`label`	string	Required. The API label of the custom feed filter.
`isProtected`	boolean	An auto-generated value. It currently has no impact.

FeedFilterCriterion

Represents the conditions that a feed item must satisfy to be displayed when a feed filter is applied.

Field Name	Field Type	Description
`feedItemType`	FeedItemType (enumeration of type string)	Required. The type of feed items that the filter shows. The feed item type can be one of the following values: • AttachArticleEvent • CallLogPost • CanvasPost • CaseCommentPost • ChangeStatusPost • ChatTranscriptPost • ContentPost • CreateRecordEvent • EmailMessageEvent

Field Name	Field Type	Description
		• LinkPost • MilestoneEvent • QuestionPost • PollPost • ReplyPost • SocialPost • TextPost
`feedItemVisibility`	FeedItemVisibility (enumeration of type string)	The visibility of feed items that the filter shows. For example, you can show only poll posts that are visible internally. Valid values are: • AllUsers • InternalUsers
`relatedSObjectType`	string	The API name of the object that the feed item refers to. This field is typically used with the CreateRecordEvent feed item type. For example, a feed filter can show CreateRecordEvent feed items for the Cases object.

Declarative Metadata Sample Definition

The following is an example of a CustomFeedFilter component.

```
<?xml version="1.0" encoding="UTF-8"?>
<CustomFeedFilter xmlns="http://soap.sforce.com/2006/04/metadata">
    <criteria>
        <feedItemType>CreateRecordEvent</feedItemType>
        <relatedSObjectType>MyCO01__c</relatedSObjectType>
    </criteria>
    <criteria>
        <feedItemType>CreateRecordEvent</feedItemType>
        <relatedSObjectType>Case</relatedSObjectType>
    </criteria>
    <criteria>
        <feedItemType>PollPost</feedItemType>
        <feedItemVisibility>InternalUsers</feedItemVisibility>
    </criteria>
    <label>Sample Custom Feed Filter</label>
</CustomFeedFilter>
```

The following is an example `package.xml` that references the previous definition.

```
<?xml version="1.0" encoding="UTF-8"?>
<Package xmlns="http://soap.sforce.com/2006/04/metadata">
```

```
    <types>
        <members>myCaseFeedFilter</members>
        <name>CustomFeedFilter</name>
    </types>
    <version>48.0</version>
</Package>
```

Wildcard Support in the Manifest File

This metadata type supports the wildcard character `*` (asterisk) in the `package.xml` manifest file. For information about using the manifest file, see Deploying and Retrieving Metadata with the Zip File.

CustomHelpMenuSection

Represents the section of the Lightning Experience help menu that the admin added to display custom, org-specific help resources for the org. The custom section contains help resources added by the admin. This type extends the Metadata metadata type and inherits its `fullName` field.

File Suffix and Directory Location

CustomHelpMenuSection components have the suffix `.customHelpMenuSection` and are stored in the `customHelpMenuSections` folder.

Version

CustomHelpMenuSection components are available in API version 45.0 and later.

Fields

Field Name	Field Type	Description
`customHelpMenuItems`	CustomHelpMenuItems[]	Items included in the custom section. Specify up to 15 items.
`masterLabel`	string	Required. Name of the custom section. Only one custom section can be added to the Lightning Experience help menu. Specify up to 80 characters.

CustomHelpMenuItems

Items included in the custom section. Specify up to 15 items.

Field Name	Field Type	Description
`linkURL`	string	Required. The URL for the resource.
`masterLabel`	string	Required. The name of the resource. Specify up to 100 characters.

Field Name	Field Type	Description
sortOrder	int	Required. The order of the item within the custom section. Valid values are 1 through 15.

Declarative Metadata Sample Definition

The following is an example of a CustomHelpMenuSection component.

```xml
<?xml version="1.0" encoding="UTF-8"?>
<CustomHelpMenuSection xmlns="http://soap.sforce.com/2006/04/metadata">
    <masterLabel>MyOrgCustomHelp</masterLabel>
    <customHelpMenuItems>
        <linkUrl>https://www.yourcompanyhelp.com/gettingstarted</linkUrl>
        <masterLabel>Getting Started</masterLabel>
        <sortOrder>1</sortOrder>
    </customHelpMenuItems>
    <customHelpMenuItems>
        <linkUrl>https://www.yourcompanyhelp.com/features</linkUrl>
        <masterLabel>Feature to Start Using Right Away</masterLabel>
        <sortOrder>2</sortOrder>
    </customHelpMenuItems>
    <customHelpMenuItems>
        <linkUrl>https://www.yourcompanyhelp.com/salestips</linkUrl>
        <masterLabel>Tips for Sales Team Members</masterLabel>
        <sortOrder>3</sortOrder>
    </customHelpMenuItems>
</CustomHelpMenuSection>
```

The following is an example package.xml that references the previous definition.

```xml
<?xml version="1.0" encoding="UTF-8"?>
<Package xmlns="http://soap.sforce.com/2006/04/metadata">
    <types>
        <members>MyOrgCustomHelp</members>
        <name>CustomHelpMenuSection</name>
    </types>
    <version>45.0</version>
</Package>
```

CustomLabels

The CustomLabels metadata type allows you to create custom labels that can be localized for use in different languages, countries, and currencies.

This type extends the Metadata metadata type and inherits its fullName field. Custom labels are custom text values, up to 1,000 characters in length, that can be accessed from Apex classes or Visualforce pages. For more information, see “Custom Labels” in Salesforce Help.

Declarative Metadata File Suffix and Directory Location

Master custom label values are stored in the `CustomLabels.labels` file. Translations for custom labels can be retrieved through Translations in Metadata API. Translations are stored in files under the `translations` folder with the name format of *`localeCode`*`.translation`, where *`localeCode`* is the locale code of the translation language. The supported locale codes are listed in Language on page 1091.

Version

CustomLabels components are available in API version 14.0 and later.

Fields

Field	Field Type	Description
`fullName`	string	Required. The name of the custom label bundle. Inherited from Metadata, this field is not defined in the WSDL for this metadata type. It must be specified when creating, updating, or deleting. See `createMetadata()` to see an example of this field specified for a call.
`labels`	CustomLabel[]	A list of custom labels.

CustomLabel

This metadata type represents a custom label. This type extends the Metadata metadata type and inherits its `fullName` field.

Field	Field Type	Description
`categories`	string	A comma-separated list of categories for the label. This field can be used in filter criteria when creating custom label list views. Maximum of 255 characters.
`fullName`	string	Required. The name of the custom label. Inherited from Metadata, this field is not defined in the WSDL for this metadata type. It must be specified when creating, updating, or deleting. See `createMetadata()` to see an example of this field specified for a call.
`language`	string	Required. The language of the translated custom label.
`protected`	boolean	Required. Indicates whether this component is protected (`true`) or not (`false`). Protected components cannot be linked to or referenced by components created in the installing organization.
`shortDescription`	string	Required. An easily recognizable term to identify this custom label. This description is used in merge fields.

Field	Field Type	Description
value	string	Required. The translated custom label. Maximum of 1000 characters.

Usage

Use CustomLabels with the wildcard character (*) for members in the `package.xml` manifest file to retrieve all custom labels that are defined in your organization. CustomLabels doesn't support retrieving one or more custom labels by name. To retrieve specific labels by name, use CustomLabel and specify the label names as members.

Declarative Metadata Sample Definition

A sample XML definition of a custom label components is shown below.

```
<?xml version="1.0" encoding="UTF-8"?>
<CustomLabels xmlns="http://soap.sforce.com/2006/04/metadata">
    <labels>
        <fullName>quoteManual</fullName>
        <value>This is a manual quote.</label>
        <language>en_US</language>
        <protected>false</protected>
        <shortDescription>Manual Quote</shortDescription>
    </labels>
    <labels>
        <fullName>quoteAuto</fullName>
        <value>This is an automatically generated quote.</label>
        <language>en_US</language>
        <protected>false</protected>
        <shortDescription>Automatic Quote</shortDescription>
    </labels>
</CustomLabels>
```

This is a sample manifest file for retrieving all custom labels in the organization by using the CustomLabels type.

```
<?xml version="1.0" encoding="UTF-8"?>
<Package xmlns="http://soap.sforce.com/2006/04/metadata">
    <fullName>MyPkg</fullName>
    <types>
      <members>*</members>
      <name>CustomLabels</name>
    </types>
    <version>48.0</version>
</Package>
```

This is a sample manifest file for retrieving two custom labels by name. Notice it uses the CustomLabel singular type.

```
<?xml version="1.0" encoding="UTF-8"?>
<Package xmlns="http://soap.sforce.com/2006/04/metadata">
    <fullName>MyPkg</fullName>
    <types>
      <members>quoteManual</members>
      <members>quoteAuto</members>
```

```
        <name>CustomLabel</name>
    </types>
    <version>48.0</version>
</Package>
```

Wildcard Support in the Manifest File

This metadata type supports the wildcard character `*` (asterisk) in the `package.xml` manifest file. For information about using the manifest file, see Deploying and Retrieving Metadata with the Zip File.

SEE ALSO:

Translations

Custom Metadata Types (CustomObject)

Represents the metadata associated with a custom metadata type.

For more information, see the *Custom Metadata Types Implementation Guide*.

File Suffix and Directory Location

A custom metadata type is defined as a custom object and is stored in the objects folder. Custom metadata types have a suffix of `__mdt` (instead of `__c` for custom objects). Custom metadata type field names have a suffix of `__c`, like other custom fields. Custom metadata type field names must be dot-qualified with the name of the custom metadata type to which they belong.

Names of custom metadata types must be unique within their namespace. All custom metadata types belong to the `CustomMetadata` namespace and can optionally belong to a second namespace. In your organization, you can use custom metadata types with your namespace and also other organizations' namespaces.

Version

Custom metadata type components are available in API version 31.0 and later.

Special Access Rules

To create custom metadata types, you must have the “Author Apex” permission. Apex code can create, read, and update (but not delete) custom metadata records, as long as the metadata is subscriber-controlled and visible from within the code's namespace. You can edit records in memory but not upsert or delete them. Apex code can deploy custom metadata records, but not via a DML operation. Moreover, DML operations aren't allowed on custom metadata in the Partner or Enterprise APIs. Customers who install a managed custom metadata type can't add new custom fields to it. With unpackaged metadata, both developer-controlled and subscriber-controlled access behave the same: like subscriber-controlled access. Refer to Trust, but Verify: Apex Metadata API and Security to learn more.

Note: Audit fields (`CreatedDate`, `CreatedBy`, `LastModifiedDate`, `LastModifiedBy`, `SystemModStamp`) remain uneditable.

Fields

Custom metadata types can contain the following CustomObject fields.

To make the fields on your custom metadata types unique and indexable, mark your fields as `Unique` and `ExternalId`.

Field Name	Field Type	Description
`description`	string	A description of the custom metadata type. This field can contain a maximum of 1,000 characters.
`fields`	CustomField[]	Represents one or more custom fields in the custom metadata type.
`gender`	Gender	Indicates the gender of the noun that represents the object. This field is used for languages where words need different treatment depending on their gender.
`label`	string	A label that represents the object throughout the Salesforce Setup user interface. Custom metadata types are visible only through the recently used objects list on the Lightning Platform Home Page and in the packaging user interface.
`pluralLabel`	string	The plural version of the label value.
`startsWith`	StartsWith (enumeration of type string)	Indicates whether the noun starts with a vowel, a consonant, or a special character. This field is used for languages where words need different treatment depending on their first character.
`visibility`	SetupObjectVisibility (enumeration of type string)	This field returns the visibility of a custom metadata type. The following values are valid: • `Public`—If the custom setting or custom metadata type is packaged, it's accessible to all subscribing organizations. • `Protected`—If the custom object, custom setting, or custom metadata type is in a managed package, it's accessible only to the developer org. Subscribing orgs can't access it. • `PackageProtected`—If the custom metadata type is `PackageProtected`, it's only accessible by the custom Apex code in the package. Use this value to secure secrets such as API access keys and security tokens. Available in API version 47.0 and later. The default value is `Public`.

Declarative Metadata Sample Definition

In this example, Picklists R Us creates its Reusable Picklist custom metadata type by deploying a file in the objects folder, named `ReusablePicklistOption__mdt.object`, with these contents.

```
<?xml version="1.0" encoding="UTF-8"?>
<CustomObject xmlns="http://soap.sforce.com/2006/04/metadata">
   <fields>
```

```
      <fullName>AlphaSort__c</fullName>
      <defaultValue>false</defaultValue>
      <externalId>false</externalId>
      <label>Sorted Alphabetically</label>
      <type>Checkbox</type>
   </fields>
   <label>Reusable Picklist</label>
   <pluralLabel>Reusable Picklist</pluralLabel>
   <visibility>Public</visibility>
</CustomObject>
```

This excerpt from a `package.xml` file shows the use of dot notation and the `__mdt` suffix. If you're using a namespace, for example `picklist1234`, the full name of `ReusablePicklistOption__mdt` would be `picklist1234__ReusablePicklistOption__mdt`.

```
<?xml version="1.0" encoding="UTF-8"?>
<Package xmlns="http://soap.sforce.com/2006/04/metadata">
...
   <types>
      <members>PicklistTest__c.PicklistTestField__c</members>
      <members>ReusablePicklistOption__mdt.Picklist__c</members>
      <members>ReusablePicklistOption__mdt.SortOrder__c</members>
      <members>PicklistUsage__mdt.Field__c</members>
      <members>PicklistUsage__mdt.Picklist__c</members>
      <members>PicklistUsage__mdt.SObjectType__c</members>
      <members>ReusablePicklist__mdt.AlphaSort__c</members>
      <name>CustomField</name>
   </types>
...
   <types>
      <members>PicklistTest__c</members>
      <members>ReusablePicklistOption__mdt</members>
      <members>PicklistUsage__mdt</members>
      <members>ReusablePicklist__mdt</members>
      <name>CustomObject</name>
   </types>
...
   <version>48.0</version>
</Package>
```

Wildcard Support in the Manifest File

This metadata type supports the wildcard character `*` (asterisk) in the `package.xml` manifest file. For information about using the manifest file, see Deploying and Retrieving Metadata with the Zip File.

IN THIS SECTION:

1. CustomMetadata

   Represents a record of a custom metadata type.

CustomMetadata

Represents a record of a custom metadata type.

This type extends the Metadata metadata type and inherits its `fullName` field.

File Suffix and Directory Location

CustomMetadata components have the suffix `.md` and are stored in the `customMetadata` folder. Unlike custom metadata types, custom metadata records don't have a double-underscore suffix. Custom metadata record names are prepended with their custom metadata type name, excluding the `__mdt` suffix but including the namespace of any types in an installed managed package.

Version

CustomMetadata components are available in API version 31.0 and later.

Special Access Rules

To create custom metadata records, you must have the "Customize Application" permission.

Fields

Field Name	Field Type	Description
`description`	string	A description of the custom metadata record. This field can contain a maximum of 1,000 characters.
`label`	string	A label that represents the object throughout the Salesforce Setup user interface. Custom metadata records are currently visible only through the packaging user interface.
`protected`	boolean	Boolean. Indicates whether the record is protected (true) or not (false). When a custom metadata type is released in a managed package, access is limited in specific ways. • Code that's in the same managed package as custom metadata records can read the records. • Code that's in the same managed package as custom metadata types can read the records that belong to that type. • Code that's in a managed package that doesn't contain either the type or the protected record can't read the protected records. • Code that the subscriber creates and code that's in an unmanaged package can't read the protected records. • The developer can modify protected records with a package upgrade or by using the Metadata Apex

Field Name	Field Type	Description
		classes (if the Apex code is in the same namespace as either the records or their type). The subscriber can't read or modify protected records. The developer name of a protected record can't be changed after release. - The subscriber can't create records of a protected type. Records that are hidden by these access rules are also unavailable to REST, SOAP, SOQL, and Setup.
`values`	CustomMetadataValue[]	Represents one or more values for custom fields on the custom metadata record.

CustomMetadataValue

Represents a value for a custom field on the custom metadata record.

Field Name	Field Type	Description
`field`	string	Required. The non-object-qualified name of a custom field in the custom metadata type. This value corresponds to the name of a field on the custom metadata record's custom metadata type. Include the namespace (if the type is from a managed package) and the `__c` suffix. The name of the custom metadata type isn't required. For example, `picklist1234__AlphaSort__c`.
`value`	Any type	The value on a custom metadata record. Where fields are EntityDefinition and FieldDefinition, the qualified API names of the entity and the field it points to. This value can be null. For more information, see Usage on page 317.

Declarative Metadata Sample Definitions

The following is an example of a CustomMetadata component. In this example, the sample app TravelApp deploys a Planets picklist, specifies its sort order, and adds picklist items to it.

Assuming Picklists R Us's namespace is `picklist1234`, to define the `Planets` picklist, TravelApp deploys a file in the `customMetadata` folder, named `picklist1234__ReusablePicklist.Planets.md`, with these contents. The `xsi:type` attribute specifies the type for the value of the `AlphaSort__c` checkbox field.

```
<?xml version="1.0" encoding="UTF-8"?>
<CustomMetadata xmlns="http://soap.sforce.com/2006/04/metadata"
                   xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
                   xmlns:xsd="http://www.w3.org/2001/XMLSchema">
    <description>All the planets in the solar system. Does not
```

```
                 include asteroids.</description>
    <label>Planets</label>
    <values>
        <field>picklist1234__AlphaSort__c</field>
        <value xsi:type="xsd:boolean">false</value>
    </values>
</CustomMetadata>
```

Picklists R Us creates its Reusable Picklist Option custom metadata type by deploying a file in the objects folder, named `ReusablePicklist__mdt.object`, with these contents.

```
<?xml version="1.0" encoding="UTF-8"?>
<CustomObject xmlns="http://soap.sforce.com/2006/04/metadata">
    <fields>
        <fullName>Picklist__c</fullName>
        <externalId>false</externalId>
        <label>Picklist</label>
        <length>40</length>
        <required>true</required>
        <type>Text</type>
        <unique>false</unique>
    </fields>
    <fields>
        <fullName>SortOrder__c</fullName>
        <externalId>false</externalId>
        <label>Non-Alphabetical Sort Order</label>
        <precision>3</precision>
        <scale>0</scale>
        <required>false</required>
        <type>Number</type>
        <unique>false</unique>
    </fields>
    <label>Reusable Picklist Option</label>
    <pluralLabel>Reusable Picklist Options</pluralLabel>
</CustomObject>
```

To define the `Mars` picklist item, TravelApp deploys a file, named `picklist1234__ReusablePicklistOption.Mars.md`, with these contents. This component file specifies types that apply to the `ReusablePicklistOption__mdt` custom fields.

```
<?xml version="1.0" encoding="UTF-8"?>
<CustomMetadata xmlns="http://soap.sforce.com/2006/04/metadata"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xmlns:xsd="http://www.w3.org/2001/XMLSchema">
    <label>Mars</label>
    <values>
        <field>picklist1234__Picklist__c</field>
        <value xsi:type="xsd:string">Planets</value>
    </values>
    <values>
        <field>picklist1234__SortOrder__c</field>
        <value xsi:type="xsd:int">4</value>
    </values>
</CustomMetadata>
```

To define the `Motel6` picklist item, TravelApp deploys a file, named `picklist1234__ReusablePicklistOption.Motel6.md`, with these contents.

```
<?xml version="1.0" encoding="UTF-8"?>
<CustomMetadata xmlns="http://soap.sforce.com/2006/04/metadata"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xmlns:xsd="http://www.w3.org/2001/XMLSchema">
    <label>Motel 6</label>
    <values>
        <field>picklist1234__Picklist__c</field>
        <value xsi:type="xsd:string">Hotels</value>
    </values>
</CustomMetadata>
```

Because the `SortOrder__c` field isn't required, this file doesn't require a value for `SortOrder__c`. Alternatively, the file could have explicitly specified a value with `xsi:nil` to ensure that `SortOrder__c` was cleared of any previous value.

```
<?xml version="1.0" encoding="UTF-8"?>
<CustomMetadata xmlns="http://soap.sforce.com/2006/04/metadata"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns:xsd="http://www.w3.org/2001/XMLSchema">
    <label>Motel 6</label>
    <values>
        <field>picklist1234__Picklist__c</field>
        <value xsi:type="xsd:string">Hotels</value>
    </values>
    <values>
        <field>picklist1234__SortOrder__c</field>
        <value xsi:nil="true" />
    </values>
</CustomMetadata>
```

This excerpt from a `package.xml` file illustrates the inclusion of custom metadata types and their namespaces in custom metadata records' names. Assume that Picklists R Us's namespace is `picklist1234`.

```
<?xml version="1.0" encoding="UTF-8"?>
<package xmlns="http://soap.sforce.com/2006/04/metadata">
…
   <types>
      <members>picklist1234__ReusablePicklist.Hotels</members>
      <members>picklist1234__ReusablePicklist.Planets</members>
      <members>picklist1234__ReusablePicklistOption.Bellagio</members>
      <members>picklist1234__ReusablePicklistOption.Motel6</members>
      <members>picklist1234__ReusablePicklistOption.Mercury</members>
      <members>picklist1234__ReusablePicklistOption.Venus</members>
      <members>picklist1234__ReusablePicklistOption.Earth</members>
      <members>picklist1234__PicklistUsage.BookedHotel</members>
      <members>
         picklist1234__PicklistUsage.DestinationPlanetPL
      </members>
      <members>picklist1234__PicklistUsage.PlanetVisitedPl</members>
      <name>CustomMetadata</name>
   </types>
…
</package>
```

TravelApp, Inc.'s `package.xml` file uses a wildcard to install custom metadata, as is shown in this excerpt from their `package.xml` file. Unless you want to deploy or retrieve specific records, using a wildcard is easier than listing all of your custom metadata records in your `package.xml` file.

```
<types>
   <members>*</members>
   <name>CustomMetadata</name>
</types>
```

If the custom metadata is from a managed package, the name after the dot in the `package.xml` file—between the two dots in the file name—is qualified by the managed package's namespace. For example, assuming TravelApp uses the namespace `travelApp1234`, the first member element in the TravelApp `package.xml` file appears to Galactic Tours as:

```
<members>picklist1234__ReusablePicklist.travelApp1234__Hotels</members>
```

Here's another example. In this case, we have an instance of custom metadata record, whose EntityDefinition field points to a custom object named `SalesAgreement__c`. The FieldDefinition field points to the custom field `CustomerReference__c` on `SalesAgreement__c`. You can deploy new custom metadata records and retrieve existing ones with EntityDefinition and FieldDefinition fields using qualified API names of custom and standard entities and their fields.

```
<?xml version="1.0" encoding="UTF-8"?><values>
<field>EntityDefintionField__c</field>
<value xsi:type="xsd:string">v1__SalesAgreement__c</value>
</values>
<values>
<field>FieldDefinitionField__c</field>
<value xsi:type="xsd:string">v1__CustomerReference__c</value>
</values>
```

Usage

When specifying the `value` field in the CustomMetadataValue subtype, specify an appropriately typed object that's based on your field type definition. In declarative metadata definitions for CustomMetadataValue, use the `xsi:type` attribute of the value element. For example, to specify a boolean value: `<value xsi:type="xsd:boolean">true</value>`. Valid `xsi:type` attributes are:

Custom metadata value	Custom field definition
`xsi:type="xsd:boolean"`	Checkbox
`xsi:type="xsd:date"`	Date
`xsi:type="xsd:dateTime"`	Date/Time
`xsi:type="xsd:picklist"`	Picklist
`xsi:type="xsd:string"`	Text
`xsi:type="xsd:string"`	Phone
`xsi:type="xsd:string"`	TextArea
`xsi:type="xsd:string"`	URL
`xsi:type="xsd:string"`	Email

Custom metadata value	Custom field definition
`xsi:type="xsd:int"`	Number/Percent, with scale equal to 0
`xsi:type="xsd:double"`	Number/Percent, with scale not equal to 0

You can also omit the `xsi:type` attribute. For example, `<value>true</value>`.

Although this attribute must be specified for any CustomMetadataValue, you can use an element with the `xsi:nil` attribute set to `true` to explicitly set the field's value to `null`. For example, `<value xsi:nil="true"/>`.

Using `null` field values differs from leaving out the CustomMetadataValue for a particular field entirely. If you leave out the CustomMetadataValue, the value of the field doesn't change. The field's value is `null` for newly deployed custom metadata records and left at its previous value for updated custom metadata records.

When you retrieve CustomMetadataValue objects, the `value` field of the returned object holds a value of the correct type, specified by `xsi:type` in the case of declarative metadata definitions.

Wildcard Support in the Manifest File

This metadata type supports the wildcard character `*` (asterisk) in the `package.xml` manifest file. For information about using the manifest file, see Deploying and Retrieving Metadata with the Zip File.

CustomNotificationType

Represents the metadata associated with a custom notification type. Custom notification types allow you to send a custom desktop or mobile notification via a process or invocable API call.

For more information about custom notifications, see Custom Notification Actions. This type extends the Metadata metadata type and inherits its `fullName` field.

Declarative Metadata File Suffix and Directory Location

The file suffix is `.notiftype` for the notification type definition. Notification types are stored in the `notificationtypes` directory of the corresponding package directory.

Version

CustomNotificationType components are available in API version 46.0 and later.

Fields

Field Name	Field Type	Description
`customNotifTypeName`	string	Required. Specifies a notification type name. Maximum number of characters: 80.
`description`	string	Specifies a general description of the notification type, which is displayed with the notification type name. Maximum number of characters: 255.

Field Name	Field Type	Description
desktop	boolean	Required. Indicates whether the desktop delivery channel is enabled (true) or not (false).
masterLabel	string	Required. Specifies the label for the notification type.
mobile	boolean	Required. Indicates whether the mobile delivery channel is enabled (true) or not (false).

Declarative Metadata Sample Definition

The following is a definition of a custom notification type.

```
<CustomNotificationType xmlns="http://soap.sforce.com/2006/04/metadata">
    <customNotifTypeName>Custom Notification</customNotifTypeName>
    <desktop>true</desktop>
    <masterLabel>Custom Notification</masterLabel>
    <mobile>true</mobile>
</CustomNotificationType>
```

Wildcard Support in the Manifest File

This metadata type doesn't support the wildcard character * (asterisk) in the package.xml manifest file. For information about using the manifest file, see Deploying and Retrieving Metadata with the Zip File.

CustomObject

Represents a custom object that stores data unique to your org or an external object that maps to data stored outside your org.This type extends the Metadata metadata type and inherits its fullName field.

Specify all relevant fields when you create or update a custom object. You cannot update a single field on the object. For more information about custom objects, see Administer Custom Objects in Salesforce Help.

You can also use this metadata type to work with customizations of standard objects, such as accounts. For an example, see Standard Objects on page 18.

All metadata components have a fullName field, which must be fully specified for any custom object.

For example, the following are fully specified names for a standard object and a custom object respectively:

```
Account
MyCustomObject__c
```

And the following is a fully specified name for an external object:

```
MyExternalObject__x
```

For sample Java code that creates a custom object, see Step 3: Walk Through the Java Sample Code on page 7.

Declarative Metadata File Suffix and Directory Location

Custom object names are automatically appended with __c. The file suffix is .object for the custom object or standard object file.

External object names are automatically appended with __x. The file suffix is `.object` for the external object file.

Custom, standard, and external objects are stored in the `objects` folder in the corresponding package directory.

Note: Retrieving a component of this metadata type in a project makes the component appear in any Profile and PermissionSet components that are retrieved in the same package.

Version

Custom objects are available in API version 10.0 and later. External objects are available in API version 32.0 and later.

Fields

Unless otherwise noted, all fields are createable, filterable, and nillable.

Field Name	Field Type	Description
`actionOverrides`	ActionOverride[]	A list of action overrides on the object. This field is available in API version 18.0 and later.
`allowInChatterGroups`	boolean	Indicates whether records of this custom object type can be added to Chatter groups. This field is available in API version 34.0 and later.
`businessProcesses`	BusinessProcess[]	A list of business processes associated with the object. This field is available in API version 17.0 and later.
`compactLayoutAssignment`	string	The compact layout assigned to the object. This field is available in API version 29.0 and later. This field is available for external objects in API version 42.0 and later.
`compactLayouts`	CompactLayout[]	A list of compact layouts associated with the object. This field is available in API version 29.0 and later. This field is available for external objects in API version 42.0 and later.
`customHelp`	string	The s-control that contains the help content if the object has customized help content. This field is available in API version 14.0 and later.
`customHelpPage`	string	The Visualforce page that contains the help content if the object has customized help content. This field is available in API version 16.0 and later.
`customSettingsType`	CustomSettingsType (enumeration of type string)	When this field is present, this component is not a custom object, but a custom setting. This field returns the type of custom setting. The following string values are valid: • `List`—static data stored in cache, accessed as part of your application, and available org-wide.

Field Name	Field Type	Description
		• Hierarchy—static data stored in cache, accessed as part of your application, and available based on a hierarchy of user, profile, or org. This value is the default. This field is available in API version 17.0 and later.
customSettingsVisibility	CustomSettingsVisibility (enumeration of type string)	When this field is present, this component is not a custom object, but a custom setting. This field returns the visibility of the custom setting. The following string values are valid: • Public—if the custom setting is packaged, it is accessible to all subscribing orgs. • Protected—if the custom setting is in a managed package, it is only accessible to the developer org. Subscribing orgs can't access it. This value is the default. This field is available in API versions 17.0 through 33.0. In versions 34.0 and later, use the visibility field instead of this field.
dataStewardGroup	string	Removed in API version 47.0.
dataStewardUser	string	Removed in API version 47.0.
deploymentStatus	DeploymentStatus (enumeration of type string)	Indicates the deployment status of the object.
deprecated	boolean	Reserved for future use.
description	string	A description of the object. Maximum of 1000 characters.
enableActivities	boolean	Indicates whether the object is enabled for activities (true) or not (false). Not available for external objects.
enableBulkApi	boolean	When enabled, the object is classified as an Enterprise Application object for usage tracking. When enabled, enableSharing and enableStreamingApi must also be enabled. This field is available in API version 31.0 and later.
enableDivisions	boolean	Indicates whether the object is enabled for divisions (true) or not (false). For more information about the Division object, see the *SOAP API Developer Guide*.
enableEnhancedLookup	boolean	Indicates whether the object is enabled for enhanced lookups (true) or not (false). In API version 28.0 and later, this field can also be used for the Account, Contact, and User objects. Enhanced lookups provide an updated lookup dialog interface that lets users filter, sort, and page through search results and customize search result columns. For more

Field Name	Field Type	Description
		information about enhanced lookups, see "Enable Enhanced Lookups" in Salesforce Help.
`enableFeeds`	boolean	Indicates whether the object is enabled for feed tracking (`true`) or not (`false`). For more information, see "Customize Chatter Feed Tracking" in Salesforce Help. This field is available in API version 18.0 and later.
`enableHistory`	boolean	Indicates whether the object is enabled for history tracking (`true`) or not (`false`). Also available for standard objects in API version 29.0 and later. History tracking on the Account object includes person account history tracking.
`enableReports`	boolean	Indicates whether the object is enabled for reports (`true`) or not (`false`). Support for external objects is available in API version 38.0 and later.
`enableSearch`	boolean	Indicates whether the object's records can be found via SOSL and Salesforce searches. Corresponds to `Allow Search` in the user interface. By default, search is disabled for new custom objects. This field is available for custom objects in API version 35.0 and later. By default, search is disabled for new external objects. However, you can validate and sync an external data source to automatically create external objects. Syncing always enables search on the external object when search is enabled on the external data source, and vice versa. This field is available for external objects in API version 37.0 and later.
`enableSharing`	boolean	When enabled, the object is classified as an Enterprise Application object for usage tracking. When enabled, `enableBulkApi` and `enableStreamingApi` must also be enabled. This field is available in API version 31.0 and later.
`enableStreamingApi`	boolean	When enabled, the object is classified as an Enterprise Application object for usage tracking. When enabled, `enableBulkApi` and `enableSharing` must also be enabled. This field is available in API version 31.0 and later.
`eventType`	PlatformEventType (enumeration of type string)	This field applies only to platform events. Indicates the event type. The values are: • `HighVolume`—For a high-volume platform event.

Field Name	Field Type	Description
		- `StandardVolume`—Deprecated. Creating a platform event with this event type is not supported and returns an error. This field is available in API version 41.0 and later.
`externalDataSource`	string	Required and available for external objects only. The name of the external data source that stores the data for the external object. The data source is represented by the ExternalDataSource component. This field is available in API version 32.0 and later.
`externalName`	string	Required and available for external objects only. The name of the table in the external data source that contains the data for the external object. This field is available in API version 32.0 and later.
`externalRepository`	string	Available for Salesforce Connect external objects only. Corresponds to `Display URL Reference Field` in the user interface. The external object's `Display URL` standard field values are automatically generated from the external system. For example, with the OData 2.0 adapter for Salesforce Connect, the value is based on the `link href` that's defined on the OData producer. You can override the default values with the values of a custom field on the same external object. Select the field name, and make sure that the custom field's values are valid URLs. This field is available in API version 32.0 and later.
`externalSharingModel`	SharingModel (enumeration of type string)	Indicates the external org-wide defaults for the object, which determines the access level for external users, such as portal and community users. You can set external org-wide defaults for these objects: - Account - Asset - Case - Campaign - Contact - Individual - Lead - Opportunity - Order - User

Field Name	Field Type	Description
		- Custom Objects Your org might have other objects whose external org-wide defaults can be modified. This field is available in API version 31.0 and later.
`fields`	CustomField[]	Represents one or more fields in the object.
`fieldSets`	FieldSet	Defines the field set that exists on this object.
`fullName`	string	Inherited from Metadata, this field is not defined in the WSDL for this metadata type. It must be specified when creating, updating, or deleting. See `createMetadata()` to see an example of this field specified for a call. This value cannot be `null`.
`gender`	Gender	Indicates the gender of the noun that represents the object. This is used for languages where words need different treatment depending on their gender.
`household`	boolean	This field supports relationship groups, a feature available only with Salesforce for Wealth Management. For more information, see “Salesforce for Wealth Management” in Salesforce Help.
`historyRetentionPolicy`	HistoryRetentionPolicy	Reserved for future use.
`indexes`	Index[]	Defines the index for a custom big object.
`label`	string	Label that represents the object throughout the Salesforce user interface. We recommend that you make object labels unique across all standard, custom, and external objects in the org.
`listViews`	ListView[]	Represents one or more *list views* associated with the object.
`namedFilter`	NamedFilter[]	Represents the metadata associated with a lookup filter. This metadata type is used to create, update, or delete lookup filter definitions. This component has been removed as of API version 30.0 and is only available in previous API versions. The metadata associated with a lookup filter is now represented by the `lookupFilter` field in the CustomField component. This field is available in API version 17.0 and later. This field has been removed as of API version 30.0 and is only available in prior versions. The metadata associated with a lookup filter is now represented by the `lookupFilter` field in the CustomField component.

Field Name	Field Type	Description
nameField	CustomField	Required for custom objects. On external objects, the name field can instead be specified by setting isNameField to true in the CustomField component. The field that this object's name is stored in. Every custom object must have a name, usually a string or autonumber. Identifier for the custom object record. This name appears in page layouts, related lists, lookup dialogs, search results, and key lists on tab home pages. By default, this field is added to the custom object page layout as a required field.
pluralLabel	string	Plural version of the label value.
profileSearchLayouts	ProfileSearchLayouts	Represents a user profile's search results layouts for an object. With profile-specific layouts, each user profile can have a different search results layout for an object. Available in API version 47.0 and later.
publishBehavior	PlatformEventPublishBehavior (enumeration of type string)	This field applies only to platform events. Indicates when platform event messages are published in a Lightning Platform transaction. This field applies to event messages published through the Lightning Platform, such as Apex, Process Builder, and Flow Builder, but not through Salesforce APIs. Valid values are: • PublishAfterCommit—The event message is published only after a transaction commits successfully. If the transaction fails, the event message isn't published. • PublishImmediately—The event message is published when the publish call executes, regardless of whether the transaction succeeds. If you don't specify this field, the default value used is PublishImmediately. This field is available in API version 46.0 and later.
recordTypes	RecordType[]	An array of one or more record types defined for this object.
recordTypeTrackFeedHistory	boolean	Indicates whether the record type is enabled for feed tracking (true) or not (false). To set this field to true, the enableFeeds field on the associated CustomObject must also be true. For more information, see "Customize Chatter Feed Tracking" in Salesforce Help. This field is available in API version 19.0 and later.
recordTypeTrackHistory	boolean	Indicates whether history tracking is enabled for this record type (true) or not (false). To set recordTypeTrackHistory to true, the

Field Name	Field Type	Description
		`enableHistory` field on the associated custom object must also be `true`. This field is available in API version 19.0 and later.
`searchLayouts`	SearchLayouts	The *Search Layouts* related list information for the object.
`sharingModel`	SharingModel(enumeration of type string)	Indicates the org-wide defaults for the object. Note: Using API version 29.0 and earlier, this field is read-only and can't be set using the Metadata API; you must use the Salesforce user interface. Using API version 30.0 and later, you can set this field for internal users using the API and the Salesforce user interface.
`sharingReasons`	SharingReason[]	The reasons why the object is being shared.
`sharingRecalculations`	SharingRecalculation[]	A list of custom sharing recalculations associated with the object.
`startsWith`	StartsWith (enumeration of type string)	Indicates whether the noun starts with a vowel, consonant, or is a special character. This is used for languages where words need different treatment depending on the first character. Valid values are listed in StartsWith.
`validationRules`	ValidationRule[]	An array of one or more validation rules on the object.
`visibility`	SetupObjectVisibility (enumeration of type string)	This field returns the visibility of the custom object, custom setting, or custom metadata type. The following values are valid. • `Public`—If the custom object, custom setting, or custom metadata type is packaged, it's accessible to all subscribing orgs. • `Protected`—If the custom object, custom setting, or custom metadata type is in a managed package, it's accessible only to the developer org. Subscribing orgs can't access it. • `PackageProtected`— (Custom metadata type only) If the custom metadata type is `PackageProtected`, it's only accessible by the custom Apex code in the package. Use this value to secure secrets such as API access keys and security tokens. Available in API version 47.0 and later. The default value is `Public`. This field is available in API version 34.0 and later. For custom settings, this field replaces the `customSettingsVisibility` field.
`webLinks`	WebLink[]	An array of one or more weblinks defined for the object.

Declarative Metadata Additional Components

CustomObject definitions can include additional components defined in the custom object for declarative metadata. The following components are defined in the CustomObject:

- ActionOverride
- BusinessProcess
- CompactLayout
- CustomField
- FieldSet
- HistoryRetentionPolicy
- ListView
- RecordType
- SearchLayouts
- SharingReason
- SharingRecalculation
- ValidationRule
- WebLink

Declarative Metadata Sample Definition

```xml
<?xml version="1.0" encoding="UTF-8"?>
<CustomObject xmlns="http://soap.sforce.com/2006/04/metadata">
    <deploymentStatus>Deployed</deploymentStatus>
    <description>test object with one field for eclipse ide testing</description>
    <fields>
        <fullName>Comments__c</fullName>
        <description>add your comments about this object here</description>
       <inlineHelpText>This field contains comments made about this object</inlineHelpText>

        <label>Comments</label>
        <length>32000</length>
        <type>LongTextArea</type>
        <visibleLines>30</visibleLines>
    </fields>
    <label>MyFirstObject</label>
    <nameField>
        <label>MyFirstObject Name</label>
        <type>Text</type>
    </nameField>
    <pluralLabel>MyFirstObjects</pluralLabel>
    <sharingModel>ReadWrite</sharingModel>
</CustomObject>
```

The following is the metadata definition of an external object for Salesforce Connect.

```xml
<?xml version="1.0" encoding="UTF-8"?>
<CustomObject xmlns="http://soap.sforce.com/2006/04/metadata">
    <actionOverrides>
        <actionName>CancelEdit</actionName>
        <type>Default</type>
    </actionOverrides>
    <actionOverrides>
        <actionName>Delete</actionName>
        <type>Default</type>
    </actionOverrides>
    <actionOverrides>
        <actionName>Edit</actionName>
        <type>Default</type>
    </actionOverrides>
    <actionOverrides>
        <actionName>Follow</actionName>
        <type>Default</type>
    </actionOverrides>
    <actionOverrides>
        <actionName>List</actionName>
        <type>Default</type>
    </actionOverrides>
    <actionOverrides>
        <actionName>New</actionName>
        <type>Default</type>
    </actionOverrides>
    <actionOverrides>
        <actionName>SaveEdit</actionName>
        <type>Default</type>
    </actionOverrides>
    <actionOverrides>
        <actionName>Tab</actionName>
        <type>Default</type>
    </actionOverrides>
    <actionOverrides>
        <actionName>View</actionName>
        <type>Default</type>
    </actionOverrides>
    <deploymentStatus>InDevelopment</deploymentStatus>
    <description>Products</description>
    <enableFeeds>false</enableFeeds>
    <externalDataSource>OData</externalDataSource>
    <externalIndexAvailable>false</externalIndexAvailable>
    <externalName>Products</externalName>
    <fields>
        <fullName>DiscontinuedDate__c</fullName>
        <description>DiscontinuedDate</description>
        <externalDeveloperName>DiscontinuedDate</externalDeveloperName>
        <externalId>false</externalId>
        <isFilteringDisabled>false</isFilteringDisabled>
        <isNameField>false</isNameField>
        <isSortingDisabled>false</isSortingDisabled>
        <label>DiscontinuedDate</label>
        <required>false</required>
        <type>DateTime</type>
    </fields>
    <fields>
        <fullName>ID__c</fullName>
        <description>ID</description>
        <externalDeveloperName>ID</externalDeveloperName>
        <externalId>false</externalId>
        <isFilteringDisabled>false</isFilteringDisabled>
        <isNameField>false</isNameField>
        <isSortingDisabled>false</isSortingDisabled>
        <label>ID</label>
        <precision>18</precision>
        <required>false</required>
        <scale>0</scale>
        <type>Number</type>
        <unique>false</unique>
    </fields>
    <fields>
        <fullName>Name__c</fullName>
        <description>Name</description>
        <externalDeveloperName>Name</externalDeveloperName>
        <externalId>false</externalId>
        <isFilteringDisabled>false</isFilteringDisabled>
        <isNameField>false</isNameField>
        <isSortingDisabled>false</isSortingDisabled>
        <label>Name</label>
        <length>128</length>
        <required>false</required>
        <type>Text</type>
        <unique>false</unique>
    </fields>
    <fields>
        <fullName>Price__c</fullName>
        <description>Price</description>
        <externalDeveloperName>Price</externalDeveloperName>
        <externalId>false</externalId>
        <isFilteringDisabled>false</isFilteringDisabled>
        <isNameField>false</isNameField>
        <isSortingDisabled>false</isSortingDisabled>
        <label>Price</label>
        <precision>16</precision>
        <required>false</required>
        <scale>2</scale>
        <type>Number</type>
        <unique>false</unique>
    </fields>
    <fields>
        <fullName>Products__c</fullName>
        <externalDeveloperName>Products</externalDeveloperName>
        <externalId>false</externalId>
        <isFilteringDisabled>false</isFilteringDisabled>
        <isNameField>false</isNameField>
        <isSortingDisabled>false</isSortingDisabled>
        <label>Products</label>
        <length>20</length>
        <referenceTo>Products__x</referenceTo>
        <relationshipLabel>Products</relationshipLabel>
        <relationshipName>Products</relationshipName>
        <type>ExternalLookup</type>
    </fields>
    <fields>
        <fullName>Rating__c</fullName>
        <description>Rating</description>
        <externalDeveloperName>Rating</externalDeveloperName>
        <externalId>false</externalId>
        <isFilteringDisabled>false</isFilteringDisabled>
        <isNameField>false</isNameField>
        <isSortingDisabled>false</isSortingDisabled>
        <label>Rating</label>
        <precision>18</precision>
        <required>false</required>
        <scale>0</scale>
        <type>Number</type>
        <unique>false</unique>
    </fields>
    <fields>
        <fullName>ReleaseDate__c</fullName>
        <description>ReleaseDate</description>
        <externalDeveloperName>ReleaseDate</externalDeveloperName>
        <externalId>false</externalId>
        <isFilteringDisabled>false</isFilteringDisabled>
        <isNameField>false</isNameField>
        <isSortingDisabled>false</isSortingDisabled>
        <label>ReleaseDate</label>
        <required>false</required>
        <type>DateTime</type>
    </fields>
    <label>Products</label>
    <pluralLabel>Products</pluralLabel>
    <searchLayouts>
        <customTabListAdditionalFields>ExternalId</customTabListAdditionalFields>
        <lookupDialogsAdditionalFields>ExternalId</lookupDialogsAdditionalFields>
       <lookupPhoneDialogsAdditionalFields>ExternalId</lookupPhoneDialogsAdditionalFields>

        <searchResultsAdditionalFields>ExternalId</searchResultsAdditionalFields>
        <searchResultsAdditionalFields>DisplayUrl</searchResultsAdditionalFields>
        <searchResultsAdditionalFields>ID__c</searchResultsAdditionalFields>
    </searchLayouts>
</CustomObject>
```

Wildcard Support in the Manifest File

This metadata type supports the wildcard character `*` (asterisk) in the `package.xml` manifest file for Field Sets and Record Types but not for other components. For information about using the manifest file, see Deploying and Retrieving Metadata with the Zip File.

IN THIS SECTION:

1. ActionOverride

   Represents an action override on a standard or custom object. Use it to create, update, edit, or delete action overrides. You can access ActionOverride only by accessing its encompassing CustomObject.

2. BusinessProcess

   The BusinessProcess metadata type enables you to display different picklist values for users based on their profile.This type extends the Metadata metadata type and inherits its `fullName` field.

3. CompactLayout

   Represents the metadata associated with a compact layout. This type extends the Metadata metadata type and inherits its `fullName` field.

4. CustomField

   Represents the metadata associated with a field. Use this metadata type to create, update, or delete custom field definitions on standard, custom, and external objects or standard field definitions on standard objects. This type extends the Metadata metadata type and inherits its `fullName` field.

5. FieldSet

   Represents a field set. A field set is a grouping of fields. For example, you could have a field set that contains fields describing a user's first name, middle name, last name, and business title.

6. HistoryRetentionPolicy

   Represents the policy for retaining field history data. By setting a policy, you can specify the number of months you want to maintain field history in Salesforce and the number of years that you want to retain field history in the archive.

7. Index

   Represents an index defined within a custom big object. Use this metadata type to define the composite primary key (index) for a custom big object.This type extends the Metadata metadata type and inherits its `fullName` field.

8. ListView

   ListView allows you to see a filtered list of records, such as contacts, accounts, or custom objects.

9. NamedFilter

   Represents the metadata associated with a lookup filter. This metadata type is used to create, update, or delete lookup filter definitions. This component has been removed as of API version 30.0 and is only available in previous API versions. The metadata associated with a lookup filter is now represented by the `lookupFilter` field in the CustomField component.

10. Picklist (Including Dependent Picklist)

    Deprecated. Represents a picklist (or dependent picklist) definition for a custom field in a custom object or a custom or standard field in a standard object, such as an account.

11. ProfileSearchLayouts

    Represents a user profile's search results layouts for an object. `ProfileSearchLayouts` are similar to `SearchLayouts`. However, with profile-specific layouts, each user profile can have a different search results layout for an object.

12. RecordType

    Represents the metadata associated with a record type. Record types let you offer different business processes, picklist values, and page layouts to different users. Use this metadata type to create, update, or delete record type definitions for a custom object.

13. SearchLayouts

    Represents the metadata associated with the Search Layouts for an object. You can customize which fields to display for users in search results, search filter fields, lookup dialogs, and recent record lists on tab home pages. You can access SearchLayouts only by accessing its encompassing CustomObject.

14. SharingReason

    Represents an Apex sharing reason, which is used to indicate why sharing was implemented for a custom object. Apex managed sharing allows developers to use Apex to programmatically share custom objects. When you use Apex managed sharing to share a custom object, only users with the “Modify All Data” permission can add or change the sharing on the custom object's record, and the sharing access is maintained across record owner changes. For more information, see “Sharing Settings” in the Salesforce online help.

15. SharingRecalculation

    Represents Apex classes that recalculate the Apex managed sharing for a specific custom object.

16. ValidationRule

    Represents a validation rule, which is used to verify that the data a user enters in a record is valid and can be saved. A validation rule contains a formula or expression that evaluates the data in one or more fields and returns a value of `true` or `false`. Validation rules also include an error message that your client application can display to the user when the rule returns a value of `true` due to invalid data.

17. WebLink

    Represents a custom button or link defined in a custom object.

18. Metadata Field Types

    These field types extend the field types described in the *SOAP API Developer Guide*.

SEE ALSO:

CustomField

Metadata

Picklist (Including Dependent Picklist)

SearchLayouts

WebLink

CustomObjectTranslation

ListView

CompactLayout

ActionOverride

Represents an action override on a standard or custom object. Use it to create, update, edit, or delete action overrides. You can access ActionOverride only by accessing its encompassing CustomObject.

Declarative Metadata File Suffix and Directory Location

Action overrides are defined as part of a standard or custom object.

Version

Action overrides are available in API version 18.0 and later. Beginning in Summer '13, action overrides can be applied to both standard and custom objects. Previously, action overrides only applied to custom objects.

Fields

Unless otherwise noted, all fields are creatable, filterable, and nillable.

Field Name	Field Type	Description
`actionName`	string	Required. The possible values are the same as the actions you can override: • `accept`

Field Name	Field Type	Description
		• `clone` • `delete` • `edit` • `list` • `new` • `tab` • `view`
`comment`	string	Any comments you want associated with the override.
`content`	string	Set this field if `type` is set to `flexipage`, `lightningcomponent`, `scontrol`, or `visualforce`. It refers to the name of the Lightning page, Lightning component, s-control, or Visualforce page to use as the override. To reference installed components, use this format: ***`Component_namespace__Component_name`***.
`formFactor`	FormFactor (enumeration of type string)	The size of the page being overridden. If the `type` field is set to `flexipage`, set this field to `Large` to override the View action with a Lightning page in Lightning Experience. The `Large` value represents the Lightning Experience desktop environment and is valid only for the `flexipage` and `lightningcomponent` types. The `Small` value represents the Salesforce app on a phone or tablet. The `Medium` value is reserved for future use. The `null` value (which is the same as specifying no value) represents Salesforce Classic. This field is available in API version 37.0 and later and is part of the feature for creating and editing record pages in Lightning Experience. Note: Lightning component overrides return different `FormFactor` values depending on the API version used. • In API version 41.0 and earlier, Lightning component overrides return only the `null` value (no value), representing the Salesforce Classic environment. • In API version 42.0, if you specify different Lightning component overrides for Lightning Experience and mobile, one component is selected randomly for both overrides and its `FormFactor` value is returned. If there's a conflict between Lightning components, and a Visualforce page override is also specified for Salesforce Classic, the Visualforce page takes precedence. • In API version 43.0 and later, a Lightning component override for Lightning Experience returns the `Large` value and a Lightning component override for mobile returns the `Small` value, as expected.

Field Name	Field Type	Description
skipRecordTypeSelect	boolean	Set this field to true if you prefer that any new records created by this action override aren't forwarded to the record type selection page. This field is only valid if the actionName is a "create" type (like new), and type is set to visualforce. This field is available in API version 21.0 and later.
type	ActionOverrideType (enumeration of type string)	Required. Represents the type of action override. Valid values are described in ActionOverrideType.

ActionOverrideType

ActionOverrideType is an enumeration of type string that defines which kind of action override to use. The valid values are:

- default—The override uses a custom override provided by an installed package. If there isn't one available, the standard Salesforce behavior is used.
- flexipage—The override uses behavior from a Lightning page, and is only valid for the View action in Lightning Experience.
- lightningcomponent—The override uses behavior from a Lightning component.
- scontrol—The override uses behavior from an s-control.
- standard—The override uses regular Salesforce behavior.
- visualforce—The override uses behavior from a Visualforce page.

Note: Existing s-controls can be used as overrides for Salesforce Classic under certain conditions. However, s-controls have been deprecated since the Spring '09 release. We recommend using Visualforce pages instead.

Declarative Metadata Sample Definitions

You can define action overrides, as in these examples for the Edit action.

A Visualforce page override for Salesforce Classic:

```
<CustomObject xmlns="http://soap.sforce.com/2006/04/metadata">
    <actionOverrides>
        <actionName>edit</actionName>
        <type>visualforce</type>
        <content>myEditVFPage</content>
        <comment>This edit action is a lot safer.</comment>
     </actionOverrides>
</CustomObject>
```

Note: This example includes no value for FormFactor. Using no value is the same as using the null value, which represents Salesforce Classic.

A Lightning component override for Lightning Experience:

```
<CustomObject xmlns="http://soap.sforce.com/2006/04/metadata">
    <actionOverrides>
        <actionName>edit</actionName>
        <type>lightningcomponent</type>
        <content>myEditLightningComponent</content>
        <formFactor>Large</formFactor>
        <comment>This edit action is a lot safer.</comment>
     </actionOverrides>
</CustomObject>
```

A Lightning component override for the Salesforce app:

```
<CustomObject xmlns="http://soap.sforce.com/2006/04/metadata">
    <actionOverrides>
        <actionName>edit</actionName>
        <type>lightningcomponent</type>
        <content>myEditLightningComponent</content>
        <formFactor>Small</formFactor>
        <comment>This edit action is a lot safer.</comment>
     </actionOverrides>
</CustomObject>
```

When overrides are included in a managed package, the overrides are represented as `default` type in the metadata. Calling `retrieve()` presents the following:

```
<CustomObject xmlns="http://soap.sforce.com/2006/04/metadata">
    <actionOverrides>
        <actionName>edit</actionName>
        <type>default</type>
     </actionOverrides>
</CustomObject>
```

If you subscribe to a managed package with default overrides, you can replace the default override behavior by editing the XML. For example, to replace the Visualforce page override with the Salesforce standard page for Salesforce Classic, use:

```
<CustomObject xmlns="http://soap.sforce.com/2006/04/metadata">
    <actionOverrides>
        <actionName>edit</actionName>
        <type>standard</type>
     </actionOverrides>
</CustomObject>
```

To set a Lightning page action override on the View standard button in Lightning Experience, use:

```
<CustomObject xmlns="http://soap.sforce.com/2006/04/metadata">
    <actionOverrides>
        <actionName>View</actionName>
        <content>myLightningPage</content>
        <formFactor>Large</formFactor>
        <type>flexipage</type>
    </actionOverrides>
</CustomObject>
```

Wildcard Support in the Manifest File

This metadata type doesn't support the wildcard character `*` (asterisk) in the `package.xml` manifest file. For information about using the manifest file, see Deploying and Retrieving Metadata with the Zip File.

SEE ALSO:

CustomObject

BusinessProcess

The BusinessProcess metadata type enables you to display different picklist values for users based on their profile.This type extends the Metadata metadata type and inherits its `fullName` field.

Multiple business processes allow you to track separate sales, support, and lead lifecycles. A sales, support, lead, or solution process is assigned to a record type. The record type determines the user profiles that are associated with the business process. For more information, see “ Managing Multiple Business Processes ” in Salesforce Help.

Declarative Metadata File Suffix and Directory Location

Business processes are defined as part of the custom object or standard object definition. See CustomObject for more information.

Version

BusinessProcess components are available in API version 17.0 and later.

Special Access Rules

Access to this object requires the View Setup and Configuration permission.

Fields

Field	Field Type	Description
`description`	string	Description for the business process.
`fullName`	string	Required. The name used as a unique identifier for API access. This field is inherited from the Metadata component, but the string it contains is created differently than the `fullName` strings for other types. For a `fullName` string BusinessProcess, the `fullName` is created combining the Entity Name and Business Process Name. For example, for a business process called “Bulk Orders” for opportunities, the `fullName` would be `Opportunity.Bulk Orders`.
`isActive`	boolean	Indicates if the business process is active (`true`) or not (`false`).
`namespacePrefix`	string	The namespace of the developer organization where the package was created.
`values`	PicklistValue[]	A list of picklist values associated with this business process.

Declarative Metadata Sample Definition

The following is a sample XML definition of a lead business process.

```
<?xml version="1.0" encoding="UTF-8"?>
<CustomObject xmlns="http://soap.sforce.com/2006/04/metadata">
....
```

```
    <businessProcesses>
        <fullName>HardwareLeadProcess</fullName>
        <description>Lead Process for hardware division</description>
        <isActive>true</isActive>
        <values>
            <fullName>Closed - Converted</fullName>
            <default>false</default>
        </values>
        <values>
            <fullName>CustomLeadStep1</fullName>
            <default>false</default>
        </values>
        <values>
            <fullName>CustomLeadStep2</fullName>
            <default>false</default>
        </values>
        <values>
            <fullName>Open - Not Contacted</fullName>
            <default>false</default>
        </values>
        <values>
            <fullName>Working - Contacted</fullName>
            <default>true</default>
        </values>
    </businessProcesses>
....
</CustomObject>
```

Wildcard Support in the Manifest File

This metadata type supports the wildcard character `*` (asterisk) in the `package.xml` manifest file only when a RecordType on page 368 is specified. For information about using the manifest file, see Deploying and Retrieving Metadata with the Zip File.

SEE ALSO:

CustomObject

CompactLayout

Represents the metadata associated with a compact layout. This type extends the Metadata metadata type and inherits its `fullName` field.

A compact layout displays a record's key fields at a glance in the Salesforce mobile app, Lightning Experience, and in the Outlook and Gmail integrations.

Compact layouts support all field types except:

- text area
- long text area
- rich text area
- multi-select picklist

For more information on compact layouts, see "Compact Layouts" in the Salesforce Help.

File Suffix and Directory Location

Compact layouts are defined as part of the custom object, standard object, or external object definition. See CustomObject for more information.

Version

CompactLayout components are available in API version 29.0 and later. CompactLayout components are available for external objects in API version 42.0 and later.

Fields

Field Name	Field Type	Description
`fields`	string	The fields assigned to the compact layout. Their order represents the prioritization given to them when defining the compact layout.
`label`	string	Label that represents the object throughout the Salesforce user interface.

Declarative Metadata Sample Definition

The following is an example of a CompactLayout component:

```
<CustomObject xmlns="http://soap.sforce.com/2006/04/metadata">
    <actionOverrides>
        <actionName>Accept</actionName>
        <type>Default</type>
    </actionOverrides>
    <actionOverrides>
        <actionName>Clone</actionName>
        <type>Default</type>
    </actionOverrides>
    <actionOverrides>
        <actionName>Delete</actionName>
        <type>Default</type>
    </actionOverrides>
    <actionOverrides>
        <actionName>Edit</actionName>
        <type>Default</type>
    </actionOverrides>
    <actionOverrides>
        <actionName>List</actionName>
        <type>Default</type>
    </actionOverrides>
    <actionOverrides>
        <actionName>New</actionName>
        <type>Default</type>
    </actionOverrides>
    <actionOverrides>
        <actionName>Tab</actionName>
        <type>Default</type>
    </actionOverrides>
    <actionOverrides>
        <actionName>View</actionName>
        <type>Default</type>
    </actionOverrides>
    <compactLayouts>
        <fullName>testCompactLayout</fullName>
        <compactLayoutItems>textfield__c</compactLayoutItems>
        <label>testCompactLayoutLabel</label>
    </compactLayouts>
    <compactLayoutAssignment>SYSTEM</compactLayoutAssignment>
    <deploymentStatus>Deployed</deploymentStatus>
    <enableActivities>false</enableActivities>
    <enableFeeds>false</enableFeeds>
    <enableHistory>false</enableHistory>
    <enableReports>false</enableReports>
    <fields>
        <fullName>textfield__c</fullName>
        <externalId>false</externalId>
        <label>textfield</label>
        <length>255</length>
        <required>false</required>
        <type>Text</type>
        <unique>false</unique>
    </fields>
    <label>customObj</label>
    <nameField>
        <label>customObj Name</label>
        <type>Text</type>
    </nameField>
    <pluralLabel>customObjs</pluralLabel>
    <recordTypes>
        <fullName>RT1</fullName>
        <active>true</active>
        <label>RT1</label>
        <compactLayoutAssignment>testCompactLayout</compactLayoutAssignment>
    </recordTypes>
    <recordTypes>
        <fullName>RT2</fullName>
        <active>true</active>
        <label>RT2</label>
    </recordTypes>
    <searchLayouts/>
    <sharingModel>ReadWrite</sharingModel>
</CustomObject>
```

Wildcard Support in the Manifest File

This metadata type supports the wildcard character `*` (asterisk) in the `package.xml` manifest file. For information about using the manifest file, see Deploying and Retrieving Metadata with the Zip File.

CustomField

Represents the metadata associated with a field. Use this metadata type to create, update, or delete custom field definitions on standard, custom, and external objects or standard field definitions on standard objects. This type extends the Metadata metadata type and inherits its `fullName` field.

Only standard fields that you can customize are supported, that is, standard fields to which you can add help text or enable history tracking or Chatter feed tracking. Other standard fields aren't supported, including system fields (such as `CreatedById` or `LastModifiedDate`) and autonumber fields.Some standard picklist fields aren't supported. See Unsupported Metadata Types.

Specify the full name whenever you create or update a field. For example, a custom field on a custom object:

```
MyCustomObject__c.MyCustomField__c
```

An example of a custom field on a standard object:

```
Account.MyAcctCustomField__c
```

An example of a standard field on a standard object:

```
Account.Phone
```

An example of a custom field on an external object:

```
MyExternalObject__x.MyCustomField__c
```

Note: In Metadata API, external objects are represented by the CustomObject metadata type.

The following custom field types aren't available for external objects.

- Auto-number (available only with the cross-org adapter for Salesforce Connect)
- Currency (available only with the cross-org adapter for Salesforce Connect)
- Formula
- Location
- Master-detail relationship
- Picklist and multi-select picklist (available only with the cross-org adapter for Salesforce Connect)
- Roll-up summary
- Text (encrypted)
- Text Area (rich)

Declarative Metadata File Suffix and Directory Location

Custom fields are user-defined fields and are part of the custom object or standard object definition. See CustomObject for more information. Standard fields are predefined on standard objects.

Note: Retrieving a component of this metadata type in a project makes the component appear in any Profile and PermissionSet components that are retrieved in the same package.

Retrieving Fields on Custom or Standard Objects

When you retrieve a custom or standard object, you return everything associated with the object, except for standard fields that aren't customizable. You can also retrieve only specific fields for an object by explicitly naming the object and fields in `package.xml`. The following definition in package.xml will create the files objects/MyCustomObject__c.object and objects/Account.object, each containing the requested field definitions.

```
<types>
  <members>MyCustomObject__c.MyCustomField__c</members>
  <members>Account.MyCustomAccountField__c</members>
  <members>Account.Phone</members>
  <name>CustomField</name>
</types>
```

Version

Custom and standard fields are available in API version 10.0 and later.

Fields

Unless otherwise noted, all fields are createable, filterable, and nillable.

Field Name	Field Type	Description
businessOwnerGroup	reference	Indicates the group associated with this field. The business owner group understands the importance of the field's data to your company, and might be responsible for determining the minimum security classification. This field is available in API version 45.0 and later.
businessOwnerUser	reference	Indicates the person associated with this field. The business owner understands the importance of the field's data to your company, and might be responsible for determining the minimum security classification. This field is available in API version 45.0 and later.
businessStatus	picklist	Indicates whether the field is in use. Valid values include: • Active • DeprecateCandidate • Hidden This field is available in API version 45.0 and later
caseSensitive	boolean	Indicates whether the field is case-sensitive (true) or not (false). For indirect lookup relationship fields on external objects, this attribute affects how this custom field's values are matched against the values of the referenceTargetField.
complianceGroup	multipicklist	Indicates the compliance acts, definitions, or regulations related to the field's data. Valid values include: • CCPA • COPPA • GDPR • HIPAA

Field Name	Field Type	Description
		- `PCI` - `PII` This field is available in API version 47.0 and later.
`customDataType`	string	Deprecated in the Spring '19 (API version 45.0) release.
`defaultValue`	string	If specified, represents the default value of the field.
`deleteConstraint`	DeleteConstraint (enumeration of type string)	Provides deletion options for lookup relationships. Valid values are: **`SetNull`** This is the default. If the lookup record is deleted, the lookup field is cleared. **`Restrict`** Prevents the record from being deleted if it's in a lookup relationship. **`Cascade`** Deletes the lookup record as well as associated lookup fields. For more information on lookup relationships, see "Object Relationships" in the Salesforce Help.
`deprecated`	boolean	Reserved for future use.
`description`	string	Description of the field.
`displayFormat`	string	The display format.
`displayLocationInDecimal`	boolean	Indicates how the geolocation values of a Location custom field appears in the user interface. If `true`, the geolocation values appear in decimal notation. If `false`, the geolocation values appear as degrees, minutes, and seconds.
`encrypted`	boolean	Note: This entry is about Shield Platform Encryption, not Classic Encryption. Indicates whether this field is encrypted (`true`) or not (`false`). This field is available in API version 34.0 through 43.0.
`encryptionScheme`	EncryptionScheme (enumeration of type string)	Note: This entry is about Shield Platform Encryption, not Classic Encryption. For encrypted fields, determines which encryption scheme a field takes. Valid values are `None`, `ProbabilisticEncryption`, `CaseSensitiveDeterministicEncryption`, and `CaseInsensitiveDeterministicEncryption`. This field is available in API version 44.0 and later.
`externalDeveloperName`	string	Available only for external objects. Name of the table column on the external data source that maps to this custom field in Salesforce.

Field Name	Field Type	Description
		Corresponds to `External Column Name` in the user interface. This field is available in API version 32.0 and later.
`externalId`	boolean	Indicates whether the field is an external ID field (`true`) or not (`false`).
`fieldManageability`	FieldManageability (enumeration of type string)	Determines who can update the field after it's released in a managed package. Valid values: • Locked—The field can't be updated. • DeveloperControlled—The creator of the record can update the field with a package upgrade. • SubscriberControlled—Anyone with proper permissions can update the field. The field can't be updated with a package upgrade. Available only for fields on custom metadata types. If the field type is `MetadataRelationship`, and the manageability of the entity definition field is: • Subscriber-controlled, then the Field Definition field must be subscriber-controlled. • Upgradeable, then the Field Definition field must be either upgradeable or subscriber-controlled.
`formula`	string	If specified, represents a formula on the field.
`formulaTreatBlankAs`	TreatBlanksAs (enumeration of type string)	Indicates how to treat blanks in a formula. Valid values are `BlankAsBlank` and `BlankAsZero`.
`fullName`	string	Inherited from Metadata, this field is not defined in the WSDL for this metadata type. It must be specified when creating, updating, or deleting. See `createMetadata()` to see an example of this field specified for a call. This value cannot be `null`.
`globalPicklist`	string.	(This field is available in API version 37.0 only and removed from later versions.) If this custom field is a picklist that's based on a global picklist, `globalPicklist` is the name of the global picklist whose value set this picklist inherits. A custom picklist that's based on a global picklist is restricted. You can only add or remove values by editing the global picklist.
`indexed`	boolean	Indicates if the field is indexed. If this field is unique or the `externalId` is set true, the `isIndexed` value is set to true. This field has been deprecated as of version 14.0 and is only provided for backward compatibility.
`inlineHelpText`	string	Represents the content of field-level help. For more information, see "Define Field-Level Help" in the Salesforce Help.

Field Name	Field Type	Description
isAIPredictionField	boolean	Available for Number type custom fields when you use Einstein Prediction Builder. Denotes whether the field can store and display Einstein prediction data on an object. Use Einstein Prediction Builder to determine the data for the target field. This field is available in API version 43.0 and later.
isFilteringDisabled	boolean	Available only for external objects. Indicates whether the custom field is available in filters. This field is available in API version 32.0 and later.
isNameField	boolean	Available only for external object fields of type text. For each external object, you can specify one field as the name field. If you set this to `true`, make sure that the external table column identified by the `externalDeveloperName` attribute contains name values. This field is available in API version 32.0 and later.
isSortingDisabled	boolean	Available only for external objects. Indicates whether the custom field is sortable. This field is available in API version 32.0 and later.
label	string	Label for the field. You cannot update the label for standard picklist fields, such as the `Industry` field for accounts.
length	int	Length of the field.
lookupFilter	LookupFilter	Represents the metadata associated with a lookup filter. This metadata type is used to create, update, or delete lookup filter definitions. This component has been removed as of API version 30.0 and is only available in previous API versions. The metadata associated with a lookup filter is now represented by the `lookupFilter` field in the CustomField component. This field is available in API version 30.0 and later. Note: LookupFilter is not supported on the article type object.
maskChar	EncryptedFieldMaskChar (enumeration of type string)	Note: This page is about Classic Encryption, not Shield Platform Encryption. What's the difference? For encrypted fields, specifies the character to be used as a mask. Valid values are enumerated in EncryptedFieldMaskChar. For more information on encrypted fields, see “Classic Encryption for Custom Fields” in the Salesforce Help.
maskType	EncryptedFieldMaskType (enumeration of type string)	Note: This page is about Classic Encryption, not Shield Platform Encryption. What's the difference? For encrypted text fields, specifies the format of the masked and unmasked characters in the field. Valid values are enumerated in

Field Name	Field Type	Description
		EncryptedFieldMaskType For more information on encrypted fields, see "Classic Encryption for Custom Fields" in the Salesforce Help.
metadataRelationship ControllingField	string	In custom metadata relationships, represents the controlling field that specifies the standard or custom object in an entity definition metadata relationship. Required when creating a field definition metadata relationship on a custom metadata type. The object specified in the controlling field determines the values available in its dependent field definition. For example, specifying the Account object filters the available fields in the field definition to Account fields only. This field is available in API version 39.0 and later.
picklist	Picklist	(**Deprecated.** Use this field in API version 37.0 and earlier only. In later versions, use valueSet instead.) If specified, the field is a picklist, and this field enumerates the picklist values and labels.
populateExistingRows	boolean	Indicates whether existing rows will be populated (true) or not (false).
precision	int	The precision for number values. Precision is the number of digits in a number. For example, the number 256.99 has a precision of 5.
referenceTargetField	string	Available only for indirect lookup relationship fields on external objects. Specifies the custom field on the parent object to match against this indirect lookup relationship field, whose values come from an external data source. The specified custom field on the parent object must have both externalId and unique set to true. This field is available in API version 32.0 and later.
referenceTo	string	If specified, indicates a reference this field has to another object.
relationshipLabel	string	Label for the relationship.
relationshipName	string	If specified, indicates the value for one-to-many relationships. For example, in the object MyObject that had a relationship to YourObject, the relationship name might be YourObjects.
relationshipOrder	int	This field is valid for all master-detail relationships, but the value is only non-zero for junction objects. A junction object has two master-detail relationships, and is analogous to an association table in a many-to-many relationship. Junction objects must define one parent object as primary (0), the other as secondary (1). The definition of primary or secondary affects delete behavior and inheritance of look and feel, and record ownership for junction objects. For more information, see the Salesforce Help. 0 or 1 are the only valid values, and 0 is always the value for objects that are not junction objects.

Field Name	Field Type	Description
reparentableMasterDetail	boolean	Indicates whether the child records in a master-detail relationship on a custom object can be reparented to different parent records. The default value is `false`. This field is available in API version 25.0 and later.
required	boolean	Indicates whether the field requires a value on creation (`true`) or not (`false`).
scale	int	The scale for the field. Scale is the number of digits to the right of the decimal point in a number. For example, the number 256.99 has a scale of 2.
securityClassification	picklist	Indicates the sensitivity of the data contained in the field. Valid values include: • `Public` • `Internal` • `Confidential` • `Restricted` • `MissionCritical` This field is available in API version 45.0 and later.
startingNumber	int	If specified, indicates the starting number for the field. When you create records, `Starting Number`'s value increments to store the number that will be assigned to the next auto-number field created. Note: • You can't retrieve the starting number of an auto-number field through Metadata API. To specify a `Starting Number` while deploying, add a `startingNumber` tag for your field to your `package.xml` file. For example: `<startingNumber>42</startingNumber>` • If you deploy without specifying a `Starting Number` value in your `package.xml` file, the default starting number for standard fields is `0`. The default starting number for custom fields is `1`.
stripMarkup	boolean	Set to `true` to remove markup, or `false` to preserve markup. Used when converting a rich text area to a long text area.
summarizedField	string	Represents the field on the detail row that is being summarized. This field cannot be null unless the `summaryOperation` value is `count`.

Field Name	Field Type	Description
summaryFilterItems	FilterItem[]	Represents the set of filter conditions for this field if it is a summary field. This field will be summed on the child if the filter conditions are met.
summaryForeignKey	string	Represents the master-detail field on the child that defines the relationship between the parent and the child.
summaryOperation	SummaryOperations (enumeration of type string)	Represents the sum operation to be performed. Valid values are enumerated in SummaryOperations.
trackFeedHistory	boolean	Indicates whether the field is enabled for feed tracking (`true`) or not (`false`). To set this field to `true`, the `enableFeeds` field on the associated CustomObject must also be `true`. For more information, see "Customize Chatter Feed Tracking" in the Salesforce Help. This field is available in API version 18.0 and later.
trackHistory	boolean	Indicates whether history tracking is enabled for the field (`true`) or not (`false`). Also available for standard object fields (picklist and lookup fields only) in API version 30.0 and later. To set `trackHistory` to `true`, the `enableHistory` field on the associated standard or custom object must also be `true`. For more information, see "Field History Tracking" in the Salesforce Help. Field history tracking isn't available for external objects.
trackTrending	boolean	Indicates whether historical trending data is captured for the field (`true`) or not (`false`). An object is enabled for historical trending if this attribute is `true` for at least one field. Available in API version 29.0 and later. For more information, see "Report on Historical Changes" in the Salesforce Help.
trueValueIndexed	boolean	This is only relevant for a checkbox field. If set, true values are built into the index. This field has been deprecated as of API version 14.0 and is only provided for backward compatibility.
type	FieldType	Indicates the field type for the field. Valid values are enumerated in FieldType. For standard fields on standard objects, the `type` field is optional. This field is included for some standard field types, such as Picklist or Lookup, but not for others. The `type` field is included for custom fields.
unique	boolean	Indicates whether the field is unique (`true`) or not (`false`).
valueSet	ValueSet	Represents the set of values that make up a picklist on a custom field. Each value is defined as a CustomValue on page 409. If this

Field Name	Field Type	Description
		custom field is a picklist that uses a global value set, `valueSet` is the name of the global value set whose values this picklist inherits. A custom picklist that uses a global value set is restricted. You can only add or remove values by editing the global value set. Note: A ValueSet component has either a `valueSetDefinition` or a `valueName` specified, but never both. This field is available in API version 38.0 and later.
`visibleLines`	int	Indicates the number of lines displayed for the field.
`writeRequiresMasterRead`	boolean	Sets the minimum sharing access level required on the master record to create, edit, or delete child records. This field applies only to master-detail or junction object custom field types. • `true`—Allows users with “Read” access to the master record permission to create, edit, or delete child records. This setting makes sharing less restrictive. • `false`—Allows users with “Read/Write” access to the master record permission to create, edit, or delete child records. This setting is more restrictive than `true`, and is the default value. For junction objects, the most restrictive access from the two parents is enforced. For example, if you set to `true` on both master-detail fields, but users have “Read” access to one master record and “Read/Write” access to the other master record, users won't be able to create, edit, or delete child records.

Fields use additional data types. For more information, see Metadata Field Types on page 381.

EncryptedFieldMaskChar

This field type is used in `maskChar`. It is a string with two valid values: `asterisk` or `X`. For more information on encrypted fields, see Classic Encryption for Custom Fields in the Salesforce online help.

EncryptedFieldMaskType

This field type is used in `maskType`. Valid values are:

`all`

All characters in the field are hidden. This option is equivalent to the `Mask All Characters` option in Salesforce.

`creditCard`

The first 12 characters are hidden and the last four display. This option is equivalent to the `Credit Card Number` option in Salesforce.

`ssn`
The first five characters are hidden and the last four display. This option is equivalent to the `Social Security Number` option in Salesforce.

`lastFour`
All characters are hidden but the last four display. This option is equivalent to the `Last Four Characters Clear` option in Salesforce.

`sin`
All characters are hidden but the last four display. This option is equivalent to the `Social Insurance Number` option in Salesforce.

`nino`
All characters are hidden. Salesforce automatically inserts spaces after each pair of characters if the field contains nine characters. This option is equivalent to the `National Insurance Number` option in Salesforce.

For more information on encrypted fields, see Classic Encryption for Custom Fields in the Salesforce online help.

LookupFilter

Represents the metadata associated with a lookup filter. Replaces the NamedFilter component, which was removed as of API version 30.0. LookupFilter is available in API version 30.0 and later.

Field	Field Type	Description
`active`	boolean	Required. Indicates whether or not the lookup filter is active.
`booleanFilter`	string	Specifies advanced filter conditions.
`description`	string	A description of what this filter does.
`errorMessage`	string	The error message that appears if the lookup filter fails.
`filterItems`	FilterItem[]	Required. The set of filter conditions. You can have up to 10 FilterItems per lookup filter.
`infoMessage`	string	The information message displayed on the page. Use to describe things the user might not understand, such as why certain items are excluded in the lookup filter.
`isOptional`	boolean	Required. Indicates whether or not the lookup filter is optional.

Lookup filters use additional data types. For more information, see Metadata Field Types.

FilterItem

Represents one entry in a set of filter criteria.

Field	Field Type	Description
`field`	string	Represents the field specified in the filter.

Field	Field Type	Description
`operation`	FilterOperation (enumeration of type string)	Represents the filter operation for this filter item. Valid values are enumerated in FilterOperation.
`value`	string	Represents the value of the filter item being operated upon, for example, if the filter is `my_number_field__c > 1`, the value of `value` is `1`.
`valueField`	string	Specifies if the final column in the filter contains a field or a field value. Approval processes don't support `valueField` entries in filter criteria.

FilterOperation

This is an enumeration of type string that lists different filter operations. Valid values are:

- `equals`
- `notEqual`
- `lessThan`
- `greaterThan`
- `lessOrEqual`
- `greaterOrEqual`
- `contains`
- `notContain`
- `startsWith`
- `includes`
- `excludes`
- `within` (`DISTANCE` criteria only)

SummaryOperations

Represents the type of a `summaryOperation`. Valid values are:

- `Count`
- `Min`
- `Max`
- `Sum`

Declarative Metadata Sample Definition

The following example shows a field definition for a custom field that is named `Comments__c`.

```
<?xml version="1.0" encoding="UTF-8"?>
<CustomObject xmlns="http://soap.sforce.com/2006/04/metadata">
....
```

```
<fields>
        <fullName>Comments__c</fullName>
        <description>Add your comments about this object here</description>
        <inlineHelpText>This field contains help text for this object</inlineHelpText>
        <label>Comments</label>
        <length>32000</length>
        <type>LongTextArea</type>
        <visibleLines>30</visibleLines>
</fields>
....
</CustomObject>
```

The following is the definition for two fields on the Account standard object—a custom field (`MyCustomAccountField__c`), and a standard field (`Phone`) that has history tracking enabled.

```
<?xml version="1.0" encoding="UTF-8"?>
<CustomObject xmlns="http://soap.sforce.com/2006/04/metadata">
    <fields>
        <fullName>MyCustomAccountField__c</fullName>
        <description>A custom field on the Account standard object.</description>
        <externalId>false</externalId>
        <inlineHelpText>Some help text.</inlineHelpText>
        <label>MyCustomAccountField</label>
        <length>100</length>
        <required>false</required>
        <trackFeedHistory>false</trackFeedHistory>
        <trackHistory>false</trackHistory>
        <type>Text</type>
        <unique>false</unique>
    </fields>
    <fields>
        <fullName>Phone</fullName>
        <trackFeedHistory>false</trackFeedHistory>
        <trackHistory>true</trackHistory>
    </fields>
</CustomObject>
```

Wildcard Support in the Manifest File

This metadata type doesn't support the wildcard character `*` (asterisk) in the `package.xml` manifest file. For information about using the manifest file, see Deploying and Retrieving Metadata with the Zip File.

SEE ALSO:

- CustomObject
- Picklist (Including Dependent Picklist)
- Metadata
- NamedFilter

FieldSet

Represents a field set. A field set is a grouping of fields. For example, you could have a field set that contains fields describing a user's first name, middle name, last name, and business title.

Field sets can be referenced on Visualforce pages dynamically. If the page is added to a managed package, administrators can add, remove, or reorder fields in a field set to modify the fields presented on the Visualforce page without modifying any code.

Version

FieldSet components are available in API version 21.0 and later.

Fields

Field	Field Type	Description
`availableFields`	FieldSetItem[]	An array containing all the possible fields in the field set.
`description`	string	Required. A description provided by the developer that describes the field set. This is required.
`displayedFields`	FieldSetItem[]	An array containing all the fields that are presented on the Visualforce page. The order in which a field is listed determines the order of appearance on the page.
`label`	string	Required. The label used to reference the field set.

FieldSetItem

FieldSetItem represents an individual field in a field set.

Field	Field Type	Description
`field`	string	Required. The name of a field in a standard or custom object.
`isFieldManaged`	boolean	Read-only. Denotes whether the field was added to the field set via a managed or unmanaged package.
`isRequired`	boolean	Read-only. Indicates whether the field is universally required (`true`) or not (`false`).

Declarative Metadata Sample Definition

A sample XML definition of a FieldSet component is shown below.

```
<?xml version="1.0" encoding="UTF-8"?>
<CustomObject xmlns="http://soap.sforce.com/2006/04/metadata">
    <fieldSets>
        <fullName>FieldSetNames</fullName>
        <availableFields>
            <field>MiddleName__c</field>
        </availableFields>
        <availableFields>
            <field>Title__c</field>
        </availableFields>
        <description>FieldSet containing how to properly address someone</description>
        <displayedFields>
            <field>FirstName__c</field>
        </displayedFields>
        <displayedFields>
            <field>LastName__c</field>
        </displayedFields>
        <label>FieldSet Names</label>
    </fieldSets>
</CustomObject>
```

Wildcard Support in the Manifest File

This metadata type supports the wildcard character `*` (asterisk) in the `package.xml` manifest file. For information about using the manifest file, see Deploying and Retrieving Metadata with the Zip File.

HistoryRetentionPolicy

Represents the policy for retaining field history data. By setting a policy, you can specify the number of months you want to maintain field history in Salesforce and the number of years that you want to retain field history in the archive.

This component is only available to users with the "RetainFieldHistory" permission.

Declarative Metadata File Suffix and Directory Location

Field history retention policies are defined as part of a standard or custom object. You can set field history retention policies for objects individually. See CustomObject for more information.

Version

Available in API version 31.0 and later.

Fields

Field Name	Field Type	Description
`archiveAfterMonths`	int	Required. The number of months that you want to keep field history data in Salesforce before archiving. You can set a minimum of 1 month and a maximum of 18 months. If you don't set a number, the default is 18 months. (That is, Salesforce maintains data for 18 months before archiving.)
`archiveRetentionYears`	int	Required. The number of years that you want to retain data in the archive. You can set a minimum of zero years, and a maximum of 10 years. If no number is set, the default is 10 years.

Field Name	Field Type	Description
description	string	A text description for the history retention.
gracePeriodDays	int	The number of days of extra time after the archiveAfterMonths period before the data is archived. The gracePeriodDays interval applies only to the first time that the data is archived; because all the data is copied the first time, the operation may take longer than subsequent times when only the data that changed since the last archival operation is copied. The gracePeriodDays provides extra time for the administrator to prepare the organization before the initial archive operation. You can set a minimum of zero days and a maximum of 10 days. If no number is set, the default is 1 day.

Declarative Metadata Sample Definition

This sample shows the definition of a history retention policy for a custom object:

```
<?xml version="1.0" encoding="UTF-8"?>
<CustomObject xmlns="http://soap.sforce.com/2006/04/metadata">
   <historyRetentionPolicy>
       <archiveAfterMonths>6</archiveAfterMonths>
       <archiveRetentionYears>5</archiveRetentionYears>
       <description>My field history retention</description>
   </historyRetentionPolicy>

...
</CustomObject>
```

Index

Represents an index defined within a custom big object. Use this metadata type to define the composite primary key (index) for a custom big object.This type extends the Metadata metadata type and inherits its fullName field.

File Suffix and Directory Location

Indexes are user-defined and are part of the custom object definition for big objects. See CustomObject for more information.

Version

The Index type is available in API version 41.0 and later.

Fields

Field Name	Field Type	Description
fields	IndexField[]	The definition of the fields in the index.

Field Name	Field Type	Description
`label`	string	Required. This name is used to refer to the big object in the user interface. Available in API version 41.0 and later.

IndexField

Defines which fields make up the index, their order, and sort direction. The order in which the fields are defined determines the order fields are listed in the index.

Field Name	Field Type	Description
`name`	string	Required. The API name for the field that's part of the index. This value must match the `fullName` value for the corresponding field in the fields section and be marked as required. Warning: When querying a big object record via SOQL and passing the results as arguments to the delete API, if any index field name has a leading or trailing white space, you can't delete the big object record.
`sortDirection`	string	Required. The sort direction of the field in the index. Valid values are `ASC` for ascending order and `DESC` for descending order.

Declarative Metadata Sample Definition

The following is an example of an index contained within the definition of a custom big object, `Customer_Interactions__b.object`.

```
<?xml version="1.0" encoding="UTF-8"?>
<CustomObject xmlns="http://soap.sforce.com/2006/04/metadata">

    <deploymentStatus>Deployed</deploymentStatus>

// Define the fields within the big object
    <fields>
        <fullName>Purchase__c</fullName>
        <label>Purchase</label>
        <length>16</length>
        <required>false</required>
        <type>Text</type>
        <unique>false</unique>
    </fields>

    <fields>
        <fullName>Order_Number__c</fullName>
        <label>Order Number</label>
        <length>16</length>
        <required>false</required>
        <type>Text</type>
        <unique>true</unique>
    </fields>
    <fields>
        <fullName>Platform__c</fullName>
        <label>Platform</label>
        <length>16</length>
        <required>true</required>
        <type>Text</type>
        <unique>false</unique>
    </fields>

    <fields>
        <fullName>Account__c</fullName>
        <label>User Account</label>
        <referenceTo>Account</referenceTo>
        <relationshipName>User_Account</relationshipName>
        <required>true</required>
        <type>Lookup</type>
    </fields>

    <fields>
        <fullName>Order_Date__c</fullName>
        <label>Order Date</label>
        <required>true</required>
        <type>DateTime</type>
    </fields>

// Define the index
    <indexes>
        <fullName>CustomerInteractionsIndex</fullName>
        <label>Customer Interactions Index</label>
        <fields>
            <name>Account__c</name>
            <sortDirection>DESC</sortDirection>
        </fields>
        <fields>
            <name>Platform__c</name>
            <sortDirection>ASC</sortDirection>
        </fields>
        <fields>
            <name>Order_Date__c</name>
            <sortDirection>DESC</sortDirection>
        </fields>
    </indexes>

    <label>Customer Interaction</label>
    <pluralLabel>Customer Interactions</pluralLabel>
</CustomObject>
```

Wildcard Support in the Manifest File

This metadata type doesn't support the wildcard character `*` (asterisk) in the `package.xml` manifest file. For information about using the manifest file, see Deploying and Retrieving Metadata with the Zip File.

SEE ALSO:

- CustomObject
- Metadata

ListView

ListView allows you to see a filtered list of records, such as contacts, accounts, or custom objects.

This type extends the Metadata metadata type and inherits its `fullName` field. See "Create a Custom List View in Salesforce Classic" in the Salesforce online help.

Note: List views with the Visible only to me `Restrict Visibility` option are not accessible in Metadata API. Each of these list views is associated with a particular user.

Declarative Metadata File Suffix and Directory Location

List views are stored within a CustomObject component. The component can represent a custom object or a standard object, such as an account.

Version

ListView components for custom objects are available in API version 14.0 and later. ListView components for standard objects, such as accounts, are available in API version 17.0 and later.

Fields

Field	Field Type	Description
`booleanFilter`	string	This field represents an Advanced Option for a filter. Advanced Options in filters allow you to build up filtering conditions that use a mixture of AND and OR boolean operators across multiple filter line items. For example, `(1 AND 2) OR 3` finds records that match both the first two filter line items or the third.
`columns`	string[]	The list of fields in the list view. The field name relative to the object name, for example *`MyCustomField__c`*, is specified for each custom field. Note: Field names in the ListView columns don't always match their API name counterparts. In particular, if person accounts is enabled in your organization, standard fields merged from a contact into an account start with the `PC_` prefix, while the corresponding API name starts with the `Person` prefix. For example, the ListView

Field	Field Type	Description
		column name is `PC_Email` for a corresponding API field name of `PersonEmail`.
`division`	string	If your organization uses divisions to segment data and you have the “Affected by Divisions” permission, records in the list view must match this division. This field is only available if you are searching all records. This field is available in API version 17.0 and later.
`filterScope`	FilterScope (enumeration of type string)	Required. This field indicates whether you are filtering by owner or viewing all records.
`filters`	ListViewFilter[]	The list of filter line items.
`fullName`	string	Required. Inherited from Metadata, this field is not defined in the WSDL for this metadata type. It must be specified when creating, updating, or deleting. See `createMetadata()` to see an example of this field specified for a call.
`label`	string	Required. The list view name.
`language`	Language	The language used for filtering if your organization uses the Translation Workbench and you are using the `startsWith` or `contains` operator. The values entered as search terms must be in the same language as the filter language. For a list of valid language values, see Language. This field is available in API version 17.0 and later.
`queue`	string	The name of a queue. Objects are sometimes assigned to a queue so that the users who have access to the queue can monitor and manage them. When you create a queue, a corresponding list view is automatically created. See “Create Queues” in the Salesforce online help.
`sharedTo`	SharedTo	Sharing access for the list view. This field is available in API version 17.0 and later.

ListViewFilter

ListViewFilter represents a filter line item.

Field	Field Type	Description
`filter`	string	Required. Represents the field specified in the filter.
`operation`	FilterOperation (enumeration of type string)	Required. The operation used by the filter, such as `equals`. The valid values are listed in FilterOperation.

Field	Field Type	Description
value	string	Represents the value of the filter item being operated upon, for example, if the filter is my_number_field__c > 1, the value of value is 1.

FilterScope

This is an enumeration of type string that represents the filtering criteria for the records. The valid values are listed in the table below:

Enumeration Value	Description
Everything	All records, for example All Opportunities.
Mine	Records owned by the user running the list view, for example My Opportunities.
MineAndMyGroups	Records owned by the user running the list view, and records assigned to the user's queues.
Queue	Records assigned to a queue.
Delegated	Records delegated to another user for action: for example, a delegated task. This option is available in API version 17.0 and later.
MyTerritory	Records in the territory of the user seeing the list view. This option is available if territory management is enabled for your organization. Opportunities can't be filtered by MyTerritory. This option is available in API version 17.0 and later.
MyTeamTerritory	Records in the territory of the team of the user seeing the list view. This option is available if territory management is enabled for your organization. Opportunities can't be filtered by MyTeamTerritory. This option is available in API version 17.0 and later.
Team	Records assigned to a team. This option is available in API version 17.0 and later.

Declarative Metadata Sample Definition

A sample XML definition of a list view in a custom object is shown below.

```
<?xml version="1.0" encoding="UTF-8"?>
<CustomObject xmlns="http://soap.sforce.com/2006/04/metadata">
. . .
    <listViews>
        <fullName>All_Mileages</fullName>
        <filterScope>everything</filterScope>
        <label>All Mileages</label>
    </listViews>
    <listViews>
        <fullName>My_Mileages</fullName>
        <booleanFilter>1 AND 2</booleanFilter>
        <columns>NAME</columns>
        <columns>CREATED_DATE</columns>
        <filterScope>mine</filterScope>
        <filters>
            <field>NAME</field>
            <operation>equals</operation>
            <value>Eric Bristow</value>
        </filters>
        <filters>
            <field>City__c</field>
            <operation>equals</operation>
            <value>Paris</value>
        </filters>
        <label>My Mileages</label>
    </listViews>
. . .
</CustomObject>
```

Wildcard Support in the Manifest File

This metadata type doesn't support the wildcard character `*` (asterisk) in the `package.xml` manifest file. For information about using the manifest file, see Deploying and Retrieving Metadata with the Zip File.

SEE ALSO:

CustomObject

Sample package.xml Manifest Files

NamedFilter

Represents the metadata associated with a lookup filter. This metadata type is used to create, update, or delete lookup filter definitions. This component has been removed as of API version 30.0 and is only available in previous API versions. The metadata associated with a lookup filter is now represented by the `lookupFilter` field in the CustomField component.

This type extends the Metadata metadata type and inherits its `fullName` field. You can also use this metadata type to work with customizations of lookup filters on standard fields.

Note: The namedFilter appears as a child of the target object of the associated lookup field.

Declarative Metadata File Suffix and Directory Location

Lookup filters are defined as part of the custom object or standard object definition. See CustomObject for more information.

Note: Retrieving a component of this metadata type in a project makes the component appear in any Profile and PermissionSet components that are retrieved in the same package.

Version

Lookup filters are available in API version 17.0 and later.

Fields

Unless otherwise noted, all fields are createable, filterable, and nillable.

Field Name	Field Type	Description
`active`	boolean	Required. Indicates whether or not the lookup filter is active.
`booleanFilter`	string	Specifies advanced filter conditions.
`description`	string	A description of what this filter does.
`errorMessage`	string	The error message that appears if the lookup filter fails.
`field`	string	Required. The `fullName` of the custom or standard field associated with the lookup filter. You can associate one relationship field with each lookup filter, and vice-versa. Note: You cannot update a field associated with a lookup filter.
`filterItems`	FilterItems[]	Required. The set of filter conditions.
`infoMessage`	string	The information message displayed on the page. Use to describe things the user might not understand, such as why certain items are excluded in the lookup filter.
`fullName`	string	Inherited from Metadata, this field is not defined in the WSDL for this metadata type. It must be specified when creating, updating, or deleting. See `createMetadata()` to see an example of this field specified for a call. This value cannot be `null`.
`isOptional`	boolean	Required. Indicates whether or not the lookup filter is optional.
`name`	string	Required. The name of the lookup filter. If you create this field in the user interface, a name is automatically assigned. If you create this field through Metadata API, you must include the `name` field.
`sourceObject`	string	The object that contains the lookup field that uses this lookup filter. Set this field if the lookup filter references fields on the source object.

Lookup filters use additional data types. For more information, see Metadata Field Types.

FilterItems

FilterItems contains the following properties:

Field	Field Type	Description
`field`	string	Represents the field specified in the filter.
`operation`	FilterOperation (enumeration of type string)	Represents the filter operation for this filter item. Valid values are enumerated in FilterOperation.

Field	Field Type	Description
`value`	string	Represents the value of the filter item being operated upon, for example, if the filter is `my_number_field__c > 1`, the value of `value` is `1`.

FilterOperation

This is an enumeration of type string that lists different filter operations. Valid values are:

- `equals`
- `notEqual`
- `lessThan`
- `greaterThan`
- `lessOrEqual`
- `greaterOrEqual`
- `contains`
- `notContain`
- `startsWith`
- `includes`
- `excludes`

Declarative Metadata Sample Definition

```
<?xml version="1.0" encoding="UTF-8"?>
<CustomObject xmlns="http://soap.sforce.com/2006/04/metadata">
....
    <namedfilters>
        <fullName>nf_Acc</fullName>
        <active>true</active>
        <booleanFilter>1 OR 2</booleanFilter>
        <field>Account.lk__c</field>
        <filterItems>
            <field>Account.Phone</field>
            <operation>notEqual</operation>
            <value>x</value>
        </filterItems>
        <filterItems>
            <field>Account.Fax</field>
            <operation>notEqual</operation>
            <value>y</value>
        </filterItems>
        <name>Acc</name>
        <sourceObject>Account</sourceObject>
    </namedfilters>
....
</CustomObject>
```

Wildcard Support in the Manifest File

This metadata type doesn't support the wildcard character `*` (asterisk) in the `package.xml` manifest file. For information about using the manifest file, see Deploying and Retrieving Metadata with the Zip File.

SEE ALSO:

- CustomObject
- Picklist (Including Dependent Picklist)
- Metadata
- CustomField

Picklist (Including Dependent Picklist)

Deprecated. Represents a picklist (or dependent picklist) definition for a custom field in a custom object or a custom or standard field in a standard object, such as an account.

Version

Use this type in API version 37.0 and earlier only. Picklists for custom fields in custom objects are available in API version 12.0 and later. Picklists for custom or standard fields in standard objects, such as accounts, are available in API version 16.0 and later.

In API version 38.0 and later, Picklist is replaced by ValueSet on the CustomField type.

Declarative Metadata File Suffix and Directory Location

Picklist definitions are included in the custom object and field with which they are associated.

Fields

Picklist contains the following fields:

Field Name	Field Type	Description
`controllingField`	string	The `fullName` of the controlling field if this is a dependent picklist. A dependent picklist works in conjunction with a controlling picklist or checkbox to filter the available options. The value chosen in the controlling field affects the values available in the dependent field. This field is available in API version 14.0 and later.
`picklistValues`	PicklistValue[]	Required. Represents a set of values for a picklist.
`restrictedPicklist`	boolean	Indicates whether the picklist's value list is restricted. With a restricted picklist, only an admin can add or change values; users can't load or remove values through the API. By default this value is `false`. This field is available in API version 37.0 and later.
`sorted`	boolean	Indicates whether values should be sorted (`true`), or not (`false`). By default this value is `false`.

Java Sample

The following sample uses a picklist. For a complete sample of using a picklist with record types and profiles, see Profile on page 731.

```
public void setPicklistValues() {
  // Create a picklist
  Picklist expenseStatus = new Picklist();
  PicklistValue unsubmitted = new PicklistValue();
  unsubmitted.setFullName("Unsubmitted");
  PicklistValue submitted = new PicklistValue();
  submitted.setFullName("Submitted");
  PicklistValue approved = new PicklistValue();
  approved.setFullName("Approved");
  PicklistValue rejected = new PicklistValue();
  rejected.setFullName("Rejected");
  expenseStatus.setPicklistValues(new PicklistValue[]
      {unsubmitted, submitted, approved, rejected});

  CustomField expenseStatusField = new CustomField();
  expenseStatusField.setFullName(
      "ExpenseReport__c.ExpenseStatus__c");
  expenseStatusField.setLabel("Expense Report Status");
  expenseStatusField.setType(FieldType.Picklist);
  expenseStatusField.setPicklist(expenseStatus);
  try {
    AsyncResult[] ars =
    metadataConnection.create(new Metadata[] {expenseStatusField});
  } catch (ConnectionException ce) {
    ce.printStackTrace();
  }
}
```

Declarative Metadata Sample Definition

The following sample shows usage for picklists, including dependent picklists, in a custom object. The `isAmerican__c` checkbox controls the list of manufacturers shown in the `manufacturer__c` picklist. The `manufacturer__c` checkbox in turn controls the list of models shown in the `model__c` picklist.

```
<?xml version="1.0" encoding="UTF-8"?>
<CustomObject xmlns="http://soap.sforce.com/2006/04/metadata">
    <deploymentStatus>Deployed</deploymentStatus>
    <enableActivities>true</enableActivities>
    <fields>
        <fullName>isAmerican__c</fullName>
        <defaultValue>false</defaultValue>
        <label>American Only</label>
        <type>Checkbox</type>
    </fields>
    <fields>
        <fullName>manufacturer__c</fullName>
        <label>Manufacturer</label>
        <picklist>
            <controllingField>isAmerican__c</controllingField>
            <picklistValues>
```

```
                <fullName>Chrysler</fullName>
                <controllingFieldValues>checked</controllingFieldValues>
                <default>false</default>
            </picklistValues>
            <picklistValues>
                <fullName>Ford</fullName>
                <controllingFieldValues>checked</controllingFieldValues>
                <default>false</default>
            </picklistValues>
            <picklistValues>
                <fullName>Honda</fullName>
                <controllingFieldValues>unchecked</controllingFieldValues>
                <default>false</default>
            </picklistValues>
            <picklistValues>
                <fullName>Toyota</fullName>
                <controllingFieldValues>unchecked</controllingFieldValues>
                <default>false</default>
            </picklistValues>
            <sorted>false</sorted>
        </picklist>
        <type>Picklist</type>
    </fields>
    <fields>
        <fullName>model__c</fullName>
        <label>Model</label>
        <picklist>
            <controllingField>manufacturer__c</controllingField>
            <picklistValues>
                <fullName>Mustang</fullName>
                <controllingFieldValues>Ford</controllingFieldValues>
                <default>false</default>
            </picklistValues>
            <picklistValues>
                <fullName>Taurus</fullName>
                <controllingFieldValues>Ford</controllingFieldValues>
                <default>false</default>
            </picklistValues>
            <picklistValues>
                <fullName>PT Cruiser</fullName>
                <controllingFieldValues>Chrysler</controllingFieldValues>
                <default>false</default>
            </picklistValues>
            <picklistValues>
                <fullName>Pacifica</fullName>
                <controllingFieldValues>Chrysler</controllingFieldValues>
                <default>false</default>
            </picklistValues>
            <picklistValues>
                <fullName>Accord</fullName>
                <controllingFieldValues>Honda</controllingFieldValues>
                <default>false</default>
            </picklistValues>
            <picklistValues>
```

```
                <fullName>Civic</fullName>
                <controllingFieldValues>Honda</controllingFieldValues>
                <default>false</default>
            </picklistValues>
            <picklistValues>
                <fullName>Prius</fullName>
                <controllingFieldValues>Toyota</controllingFieldValues>
                <default>false</default>
            </picklistValues>
            <picklistValues>
                <fullName>Camry</fullName>
                <controllingFieldValues>Toyota</controllingFieldValues>
                <default>false</default>
            </picklistValues>
            <sorted>false</sorted>
        </picklist>
        <type>Picklist</type>
    </fields>
....
</CustomObject>
```

The following sample shows usage for the standard `Stage` field in opportunities.

```
<?xml version="1.0" encoding="UTF-8"?>
<CustomObject xmlns="http://soap.sforce.com/2006/04/metadata">
    <fields>
        <fullName>StageName</fullName>
        <picklist>
            <picklistValues>
                <fullName>Prospecting</fullName>
                <default>false</default>
                <forecastCategory>Pipeline</forecastCategory>
                <probability>10</probability>
            </picklistValues>
            <picklistValues>
                <fullName>Qualification</fullName>
                <default>false</default>
                <forecastCategory>Pipeline</forecastCategory>
                <probability>10</probability>
            </picklistValues>
            <picklistValues>
                <fullName>Needs Analysis</fullName>
                <default>false</default>
                <forecastCategory>Pipeline</forecastCategory>
                <probability>20</probability>
            </picklistValues>
            ...
        </picklist>
    </fields>
<CustomObject>
```

Wildcard Support in the Manifest File

This metadata type doesn't support the wildcard character `*` (asterisk) in the `package.xml` manifest file. For information about using the manifest file, see Deploying and Retrieving Metadata with the Zip File.

ProfileSearchLayouts

Represents a user profile's search results layouts for an object. `ProfileSearchLayouts` are similar to `SearchLayouts`. However, with profile-specific layouts, each user profile can have a different search results layout for an object.

File Suffix and Directory Location

Profile search layouts are defined as part of a standard or custom object. `SearchLayout` is the default search results layout used when no layout is specified for a user profile. For more information, see CustomObject.

Version

Profile search layouts for custom objects are available in API version 48.0 and later.

Fields

Field	Field Type	Description
`profileName`	string[]	The name of the profile associated with a customized search results layout. The profile name can be a standard Salesforce profile or custom profile defined in your org.
`fields`	string[]	The list of fields displayed in search results for the object and for the users that have the profile *`Profile Name`*. The `name` field is required and is always displayed as the first column header, so it is not included in this list. All additional fields are included. The field name relative to the object name, for *`exampleMyCustomField__c`*, is specified for each custom field.

Declarative Metadata Sample Definition

The following shows a sample definition of profile-specific search layouts in an object.

Note: To deploy a profile-specific search results layout, the profile must be defined in the destination org.

```
<?xml version="1.0" encoding="UTF-8"?>
                <CustomObject xmlns="http://soap.sforce.com/2006/04/metadata">
                . . .
                <profileSearchLayouts>
                <fields>ACCOUNT.NAME</fields>
                <fields>ACCOUNT.SITE</fields>
                <fields>ACCOUNT.PHONE1</fields>
                <fields>CORE.USERS.ALIAS</fields>
```

```
<fields>ACCOUNT.ADDRESS2_CITY</fields>
<profileName>System Administrator</profileName>
</profileSearchLayouts>
<profileSearchLayouts>
<fields>ACCOUNT.NAME</fields>
<fields>ACCOUNT.SITE</fields>
<profileName>WDC Only User</profileName>
</profileSearchLayouts>
. . .
</CustomObject>
```

SEE ALSO:

SearchLayouts

RecordType

Represents the metadata associated with a record type. Record types let you offer different business processes, picklist values, and page layouts to different users. Use this metadata type to create, update, or delete record type definitions for a custom object.

For more information, see "Tailor Business Processes to Different Users" in the Salesforce Help. This type extends the Metadata metadata type and inherits its `fullName` field.

When an object is accessible to a user, all record type information for the object is readable by that user. We strongly recommend against storing sensitive information in the record type description, name, or label. Instead, store sensitive information in a separate object or fields to which you've applied appropriate access controls.

Note: Retrieving a component of this metadata type in a project makes the component appear in any Profile and PermissionSet components that are retrieved in the same package.

Version

Record types are available in API version 12.0 and later.

Fields

Field	Field Type	Description
`active`	boolean	Required. Indicates whether the record type is active.
`businessProcess`	string	The `fullName` of the business process associated with the record type. This field is required in record types for lead, opportunity, solution, and case, and not allowed otherwise. See BusinessProcess on page 336. This field is available in API version 17.0 and later.
`compactLayoutAssignment`	string	Represents the compact layout that is assigned to the record type. This field is available in API version 29.0 and later.
`description`	string	Record type description. Maximum of 255 characters.

Field	Field Type	Description
fullName	string	Record type name. The fullName can contain only underscores and alphanumeric characters. It must be unique, begin with a letter, not include spaces, not end with an underscore, and not contain two consecutive underscores. If this field contained characters before version 14.0 that are no longer allowed, the characters were stripped out of this field, and the previous value of the field was saved in the label field. Inherited from the Metadata component, this field is not defined in the WSDL for this component. It must be specified when creating, updating, or deleting. See create() to see an example of this field specified for a call. This value cannot be null.
label	string	Required. Descriptive label for the record type. The list of characters allowed in the fullName field has been reduced for versions 14.0 and later. This field contains the value contained in the fullName field before version 14.0.
picklistValues	RecordTypePicklistValue[]	Represents a set of values for a picklist.

RecordTypePicklistValue

RecordTypePicklistValue represents the combination of picklists and valid values that define a record type:

Field Name	Field Type	Description
picklist	string	Required. The name of the picklist.
values	PicklistValue	One or more of the picklist values in the picklist. Each value defined is available in the record type that contains this component.

Java Sample

The following sample uses two record types. For the complete sample that includes profiles and picklists, see Profile on page 731.

```
public void recordTypeSample() {
  try {
    // Employees and managers have different access
    // to the state of the expense sheet
    RecordType edit = new RecordType();
    edit.setFullName("ExpenseReport__c.Edit");
    edit.setLabel("ExpenseReport__c.Label");
    PicklistValue unsubmitted = new PicklistValue();
    unsubmitted.setFullName("Unsubmitted");
    PicklistValue submitted = new PicklistValue();
    submitted.setFullName("Submitted");
    RecordTypePicklistValue editStatuses =
        new RecordTypePicklistValue();
    editStatuses.setPicklist("ExpenseStatus__c");
    editStatuses.setValues(
        new PicklistValue[] {unsubmitted, submitted});
    edit.setPicklistValues(
        new RecordTypePicklistValue[] {editStatuses});
    AsyncResult[] arsEdit =
        metadataConnection.create(new Metadata[] {edit});

    RecordType approve = new RecordType();
    approve.setFullName("ExpenseReport__c.Approve");
    PicklistValue approved = new PicklistValue();
    approved.setFullName("Approved");
    PicklistValue rejected = new PicklistValue();
    rejected.setFullName("Rejected");
    RecordTypePicklistValue approveStatuses =
        new RecordTypePicklistValue();
    approveStatuses.setPicklist("ExpenseStatus__c");
    approveStatuses.setValues(
        new PicklistValue[] {approved, rejected});
    approve.setPicklistValues(
        new RecordTypePicklistValue[] {approveStatuses});
    AsyncResult[] arsApprove =
      metadataConnection.create(new Metadata[] {approve});
  } catch (ConnectionException ce) {
    ce.printStackTrace();
  }
}
```

Declarative Metadata Sample Definition

The definition of a record type in a custom object is shown below:

```
<CustomObject xmlns="http://soap.sforce.com/2006/04/metadata">
. . .
   <recordTypes>
        <fullName>My First Recordtype</fullName>
    </recordTypes>
 . . .
</CustomObject>
```

Wildcard Support in the Manifest File

This metadata type doesn’t support the wildcard character * (asterisk) in the package.xml manifest file. For information about using the manifest file, see Deploying and Retrieving Metadata with the Zip File.

SearchLayouts

Represents the metadata associated with the Search Layouts for an object. You can customize which fields to display for users in search results, search filter fields, lookup dialogs, and recent record lists on tab home pages. You can access SearchLayouts only by accessing its encompassing CustomObject.

For more information, see "Customize Search Layouts" and "Customize Search Layouts for Custom Objects" in the Salesforce online help.

Version

Search layouts for custom objects are available in API version 14.0 and later. The ability to modify search layouts for standard objects (except events and tasks) is available in API version 27.0 and later.

Fields

Field	Field Type	Description
`customTabListAdditionalFields`	string[]	The list of fields displayed in the Recent *`Object Name`* list view on a tab associated with the object. The `name` field is mandatory and is always displayed as the first column header, so it is not included in this list; all additional fields are included. The field name relative to the object name, for example *`MyCustomField__c`*, is specified for each custom field.
`excludedStandardButtons`	string[]	The list of standard buttons excluded from the search layout.
`listViewButtons`	string[]	The list of buttons available in list views for an object. This field is equivalent to the Buttons Displayed value in the *`Object Name`* `List View` in the Search Layouts related list on the object detail page in the Salesforce user interface. For more information, see "Standard and Enhanced Lookups in Salesforce Classic" in the Salesforce online help.
`lookupDialogsAdditionalFields`	string[]	The list of fields displayed in a lookup dialog for the object. The `name` field is mandatory and is always displayed as the first column header, so it is not included in this list; all additional fields are included. The field name relative to the object name, for example *`MyCustomField__c`*, is specified for each custom field. Salesforce objects often include one or more *lookup fields* that allow users to associate two records together in a relationship. For example, a contact record includes an `Account` lookup field that represents the relationship between the contact and the organization with which the contact is associated. A lookup search dialog helps you search for the record associated with the one being edited. Lookup filter fields allow you to filter your lookup search by a customized list of fields in the object. This field is equivalent to the `Lookup Dialogs` in the Search Layouts related list on the object detail page in the application user interface. For more information, see "Standard and Enhanced Lookups in Salesforce Classic" in the Salesforce online help.

Field	Field Type	Description
lookupFilterFields	string[]	The list of fields that can be used to filter enhanced lookups for an object. Enhanced lookups are optionally enabled by your administrator. The field name relative to the object name, for example *MyCustomField__c*, is specified for each custom field. This field is equivalent to the Lookup Filter Fields in the Search Layouts related list on the object detail page in the application user interface. For more information, see "Standard and Enhanced Lookups in Salesforce Classic" in the Salesforce online help.
lookupPhoneDialogsAdditionalFields	string[]	The list of phone-related fields displayed in a lookup dialog for the object. The name field is mandatory and is always displayed as the first column header, so it is not included in this list; all additional fields are included. The field name relative to the object name, for example *MyCustomField__c*, is specified for each custom field. This list enables integration of the fields with a softphone dial pad. This field is equivalent to the Lookup Phone Dialogs in the Search Layouts related list on the object detail page in the application user interface.
massQuickActions	string[]	The list of actions that you can use to perform mass quick action on records. Use this field to add an existing create or update action. You can perform mass quick action on custom objects and all standard objects that support quick actions and have a search layout in Lightning Experience. This includes but isn't limited to cases, leads, accounts, campaigns, contacts, opportunities, and work orders.
searchFilterFields	string[]	The list of fields that can be used to filter a search for the object. The field name relative to the object name, for example *MyCustomField__c*, is specified for each custom field. This field is equivalent to the Search Filter Fields in the Search Layouts related list on the object detail page in the application user interface.
searchResultsAdditionalFields	string[]	The list of fields displayed in a search result for the object. The name field is mandatory and is always displayed as the first column header, so it is not included in this list; all additional fields are included. The field name relative to the object name, for example *MyCustomField__c*, is specified for each custom field.

Field	Field Type	Description
		This field is equivalent to the `Search Results` in the Search Layouts related list on the object detail page in the application user interface.
`searchResultsCustomButtons`	string[]	The list of custom buttons available in a search result for the object. The actions associated with the buttons can be applied to any of the records returned in the search result.

Declarative Metadata Sample Definition

A sample definition of search layouts in an object is shown below.

```
<?xml version="1.0" encoding="UTF-8"?>
                <CustomObject xmlns="http://soap.sforce.com/2006/04/metadata">
                . . .
                <searchLayouts>
                <listViewButtons>New</listViewButtons>
                <listViewButtons>Accept</listViewButtons>
                <listViewButtons>ChangeOwner</listViewButtons>
              <lookupDialogsAdditionalFields>firstQuote__c</lookupDialogsAdditionalFields>

              <lookupDialogsAdditionalFields>finalQuote__c</lookupDialogsAdditionalFields>

                <massQuickActions>Create_MQA_Contact</massQuickActions>
              <searchResultsAdditionalFields>CREATEDBY_USER</searchResultsAdditionalFields>

                </searchLayouts>
                . . .
                </CustomObject>
```

Wildcard Support in the Manifest File

This metadata type doesn't support the wildcard character `*` (asterisk) in the `package.xml` manifest file. For information about using the manifest file, see Deploying and Retrieving Metadata with the Zip File.

SEE ALSO:

CustomObject

ProfileSearchLayouts

SharingReason

Represents an Apex sharing reason, which is used to indicate why sharing was implemented for a custom object. Apex managed sharing allows developers to use Apex to programmatically share custom objects. When you use Apex managed sharing to share a custom object, only users with the "Modify All Data" permission can add or change the sharing on the custom object's record, and the sharing access is maintained across record owner changes. For more information, see "Sharing Settings" in the Salesforce online help.

Use SharingReason to create, update, or delete sharing reason definitions for a custom object. This type extends the Metadata metadata type and inherits its `fullName` field.

Version

Sharing reasons are available in API version 14.0 and later.

Fields

Field	Field Type	Description
fullName	string	Required. Sharing reason name. The __c suffix is appended to custom sharing reasons. Inherited from Metadata, this field is not defined in the WSDL for this metadata type. It must be specified when creating, updating, or deleting. See createMetadata() to see an example of this field specified for a call.
label	string	Required. Descriptive label for the sharing reason. Maximum of 40 characters.

Declarative Metadata Sample Definition

The definition of a sharing reason in a custom object:

```
<CustomObject xmlns="http://soap.sforce.com/2006/04/metadata">
. . .
    <sharingReasons>
        <fullName>recruiter__c</fullName>
        <label>Recruiter</label>
    </sharingReasons>
 . . .
</CustomObject>
```

Wildcard Support in the Manifest File

This metadata type doesn't support the wildcard character * (asterisk) in the package.xml manifest file. For information about using the manifest file, see Deploying and Retrieving Metadata with the Zip File.

SharingRecalculation

Represents Apex classes that recalculate the Apex managed sharing for a specific custom object.

For more information, see "Recalculate Apex Managed Sharing" in the Salesforce online help.

Version

Sharing recalculations are available in API version 14.0 and later.

Fields

Field	Field Type	Description
`className`	string	Required. The Apex class that recalculates the Apex sharing for a custom object. This class must implement the `Database.Batchable` interface.

Declarative Metadata Sample Definition

The definition of a sharing recalculation in a custom object:

```
<CustomObject xmlns="http://soap.sforce.com/2006/04/metadata">
. . .
    <sharingRecalculations>
        <className>RecruiterRecalculation</className>
    </sharingRecalculations>
 . . .
</CustomObject>
```

Wildcard Support in the Manifest File

This metadata type doesn’t support the wildcard character `*` (asterisk) in the `package.xml` manifest file. For information about using the manifest file, see Deploying and Retrieving Metadata with the Zip File.

ValidationRule

Represents a validation rule, which is used to verify that the data a user enters in a record is valid and can be saved. A validation rule contains a formula or expression that evaluates the data in one or more fields and returns a value of `true` or `false`. Validation rules also include an error message that your client application can display to the user when the rule returns a value of `true` due to invalid data.

This type extends the Metadata metadata type and inherits its `fullName` field.

As of API version 20.0, validation rules can't have compound fields. Examples of compound fields include addresses, first and last names, dependent picklists, and dependent lookups.

As of API version 40.0, you can use validation rules with custom metadata types.

Version

Validation rules are available in API version 12.0 and later.

Fields

Field Name	Field Type	Description
`active`	boolean	Required. Indicates whether this validation rule is active, (`true`), or not active (`false`).

Field Name	Field Type	Description
description	string	A description of the validation rule.
errorConditionFormula	string	Required. The formula defined in the validation rule. If the formula returns a value of true, an error message is displayed. See "Define Validation Rules" in the Salesforce online help.
errorDisplayField	string	The fully specified name of a field in the application. If a value is supplied, the error message appears next to the specified field. If you do not specify a value or the field isn't visible on the page layout, the value changes automatically to Top of Page.
errorMessage	string	Required. The message that appears if the validation rule fails. The message must be 255 characters or less.
fullName	string	The internal name of the object. White spaces and special characters are escaped for validity. The name must: • Contain characters, letters, or the underscore (_) character • Must start with a letter • Can't end with an underscore • Can't contain two consecutive underscore characters. Inherited from the Metadata component, this field is not defined in the WSDL for this component. It must be specified when creating, updating, or deleting. See create() to see an example of this field specified for a call.

Declarative Metadata Sample Definition

A sample XML definition of a validation rule in a custom object is shown below.

```
<?xml version="1.0" encoding="UTF-8"?>
<CustomObject xmlns="http://soap.sforce.com/2006/04/metadata">
    <deploymentStatus>Deployed</deploymentStatus>
    <fields>
        <fullName>Mommy_Cat__c</fullName>
        <label>Mommy Cat</label>
        <referenceTo>Cat__c</referenceTo>
        <relationshipName>Cats</relationshipName>
        <type>Lookup</type>
    </fields>
    <label>Cat</label>
    <nameField>
        <label>Cat Name</label>
        <type>Text</type>
    </nameField>
    <pluralLabel>Cats</pluralLabel>
    <sharingModel>ReadWrite</sharingModel>
    <validationRules>
        <fullName>CatsRule</fullName>
        <active>true</active>
        <errorConditionFormula>OR(Name = &apos;Milo&apos;,Name =
&apos;Moop&apos;)</errorConditionFormula>
        <validationMessage>Name must be that of one of my cats</validationMessage>
    </validationRules>
</CustomObject>
```

Wildcard Support in the Manifest File

This metadata type doesn't support the wildcard character `*` (asterisk) in the `package.xml` manifest file. For information about using the manifest file, see Deploying and Retrieving Metadata with the Zip File.

WebLink

Represents a custom button or link defined in a custom object.

This type extends the Metadata metadata type and inherits its `fullName` field.

Version

WebLinks are available in API version 12.0 and later.

Fields

Field Name	Field Type	Description
`availability`	WebLinkAvailability (enumeration of type string)	Required. Indicates whether the button or link is only available online (`online`, or if it is also available offline (`offline`).
`description`	string	A description of the button or link.
`displayType`	WebLinkDisplayType (enumeration of type string)	Represents how the button or link is rendered. Valid values are: • `link` for a hyperlink • `button` for a button • `massActionButton` for a button attached to a related list
`encodingKey`	Encoding (enumeration of type string)	Required. The default encoding setting is Unicode: `UTF-8`. Change it if your template requires data in a different format. This is available if your content source is URL. Valid values include: • `UTF-8`—Unicode (UTF-8) • `ISO-8859-1`—General US & Western Europe (ISO-8859–1, ISO-LATIN-1) • `Shift_JIS`—Japanese (Shift-JIS) • `ISO-2022-JP`—Japanese (JIS) • `EUC-JP`—Japanese (EUC-JP) • `x-SJIS_0213`—Japanese (Shift-JIS_2004)

Field Name	Field Type	Description
		- `ks_c_5601-1987`—Korean (ks_c_5601-1987) - `Big5`—Traditional Chinese (Big5) - `GB2312`—Simplified Chinese (GB2312) - `Big5-HKSCS`—Traditional Chinese Hong Kong (Big5–HKSCS)
`fullName`	string	The name of the custom button or link with white spaces and special characters escaped for validity. The name can only contain characters, letters, and the underscore (_) character, must start with a letter, and cannot end with an underscore or contain two consecutive underscore characters. Inherited from the Metadata component, this field is not defined in the WSDL for this component. It must be specified when creating, updating, or deleting. See `create()` to see an example of this field specified for a call.
`hasMenubar`	boolean	If the `openType` is `newWindow`, this field indicates whether to show the browser menu bar for the window (`true`) or not (`false`). Otherwise, leave this field empty.
`hasScrollbars`	boolean	If the `openType` is `newWindow`, this field indicates whether to show the scroll bars for the window (`true`) or not (`false`). Otherwise, leave this field empty.
`hasToolbar`	boolean	If the `openType` is `newWindow`, this field indicates whether to show the browser toolbar for the window (`true`) or not (`false`). Otherwise, leave this field empty.
`height`	int	Height in pixels of the window opened by the custom button or link. Required if the `openType` is `newWindow`. Otherwise, leave this field empty.
`isResizable`	boolean	If the `openType` is `newWindow`, this field indicates whether to allow resizing of the window (`true`) or not (`false`). Otherwise, leave this field empty.
`linkType`	WebLinkType (enumeration of type string)	Required. Represents whether the content of the button or link is specified by a URL, an sControl, a JavaScript code block, or a Visualforce page. - `url` - `sControl` - `javascript` - `page` - `flow`—Reserved for future use.
`masterLabel`	string	Master label for this object. This display value is the internal label that is not translated.

Field Name	Field Type	Description
openType	WebLinkWindowType (enumeration of type string)	Required. When the button or link is clicked, specifies the window style that will be used to display the content. Valid values: - newWindow - sidebar - noSidebar - replace - onClickJavaScript
page	string	If the value of linkType is page, this field represents the Visualforce page. Otherwise, leave this field empty.
position	WebLinkPosition (enumeration of type string)	If the value of OpenType is newWindow, this field indicates how the new window should be displayed. Otherwise, don't specify a value. Valid values are: - fullScreen - none - topLeft
protected	boolean	Required. Indicates whether this sub-component is protected (true) or not (false). Protected sub-components can't be linked to or referenced by components or sub-components created in the installing organization.
requireRowSelection	boolean	If the openType is massAction, this field indicates whether to require individual row selection to execute the action for this button (true) or not (false). Otherwise, leave this field empty.
scontrol	string	If the value of linkType is sControl, this field represents the name of the sControl. Otherwise, leave this field empty.
showsLocation	boolean	If the openType is newWindow, this field indicates whether to show the browser location bar for the window (true) or not (false). Otherwise, leave this field empty.
showsStatus	boolean	If the openType is newWindow, this field indicates whether or not to show the browser status bar for the window. Otherwise, leave this field empty.
url	string	If the value of linkType is url, this is the URL value. If the value of linkType is javascript, this is the JavaScript content. If the value is neither of these, leave this field empty. Content must be escaped in a manner consistent with XML parsing rules.
width	int	Width in pixels of the window opened by the button or link. Required if the openType is newWindow. Otherwise, leave this field empty.

Java Sample

The following Java sample shows sample values for WebLink fields:

```
public void WebLinkSample(String name) throws Exception {
    WebLink WebLink = new WebLink();
    // name variable represents the full name of the object
    // on which to create the WebLink, for example, customObject__c
    WebLink.setFullName(name + ".googleButton");
    WebLink.setUrl("http://www.google.com");
    WebLink.setAvailability(WebLinkAvailability.online);
    WebLink.setLinkType(WebLinkType.url);
    WebLink.setEncodingKey(Encoding.fromString("UTF-8"));
    WebLink.setOpenType(WebLinkWindowType.newWindow);
    WebLink.setHeight(600);
    WebLink.setWidth(600);
    WebLink.setShowsLocation(false);
    WebLink.setHasScrollbars(true);
    WebLink.setHasToolbar(false);
    WebLink.setHasMenubar(false);
    WebLink.setShowsStatus(false);
    WebLink.setIsResizable(true);
    WebLink.setPosition(WebLinkPosition.none);
    WebLink.setMasterLabel("google");
    WebLink.setDisplayType(WebLinkDisplayType.link);

    AsyncResult[] asyncResults = metadataConnection.create(new WebLink[]{WebLink});
    // After the create() call completes, we must poll the results of checkStatus()
    //

}
```

Declarative Metadata Sample Definition

The following is the definition of a WebLink in a custom object. For related samples, see HomePageComponent and HomePageLayout.

```
<?xml version="1.0" encoding="UTF-8"?>
<CustomObject xmlns="http://soap.sforce.com/2006/04/metadata">
....
    <WebLinks>
        <fullName>googleButton</fullName>
        <availability>online</availability>
        <displayType>link</displayType>
        <encodingKey>UTF-8</encodingKey>
        <hasMenubar>false</hasMenubar>
        <hasScrollbars>true</hasScrollbars>
        <hasToolbar>false</hasToolbar>
        <height>600</height>
        <isResizable>true</isResizable>
        <linkType>url</linkType>
        <masterLabel>google</masterLabel>
        <openType>newWindow</openType>
        <position>none</position>
        <protected>false</protected>
        <showsLocation>false</showsLocation>
        <showsStatus>false</showsStatus>
        <url>http://www.google.com</url>
        <width>600</width>
    </WebLinks>
....
</CustomObject>
```

Wildcard Support in the Manifest File

This metadata type doesn't support the wildcard character `*` (asterisk) in the `package.xml` manifest file. For information about using the manifest file, see Deploying and Retrieving Metadata with the Zip File.

SEE ALSO:

HomePageComponent

HomePageLayout

CustomPageWebLink

Metadata Field Types

These field types extend the field types described in the *SOAP API Developer Guide*.

Field Type	Objects	What the Field Contains
CustomField	Custom object Custom field	Represents a custom field.
DeleteConstraint	Custom field	A string that represents deletion options for lookup relationships. Valid values are: • `SetNull` • `Restrict` • `Cascade`
DeploymentStatus	Custom object Custom field	A string which represents the deployment status of a custom object or field. Valid values are: • `InDevelopment` • `Deployed`
FieldType	Custom field	Indicates the type of a custom field. Valid values are: • `AutoNumber` • `Lookup` • `MasterDetail` • `MetadataRelationship` • `Checkbox` • `Currency`

Field Type	Objects	What the Field Contains
		• Date • DateTime • Email • EncryptedText Note: This page is about Classic Encryption, not Shield Platform Encryption. What's the difference? • ExternalLookup • IndirectLookup • Number[1] • Percent • Phone • Picklist • MultiselectPicklist • Summary • Text • TextArea • LongTextArea • Url • Hierarchy • File • Html • Location (use for geolocation fields) • Time [1] A Number custom field is internally represented as a field of type double. Setting the scale of the Number field to 0 gives you a double that behaves like an int.
Gender	Custom object	Indicates the gender of the noun that represents the object. This is used for languages where words need different treatment depending on their gender. Valid values are: • Masculine • Feminine • Neuter • AnimateMasculine (Slavic languages—currently Czech, Polish, Russian, Slovak, Slovenian, and Ukrainian) • ClassI, ClassIII, ClassV, ClassVII, ClassIX, ClassXI, ClassXIV, ClassXV, ClassXVI, ClassXVII, ClassXVIII (African languages—currently Afrikaans, Xhosa, and Zulu)

Field Type	Objects	What the Field Contains
		Note: The following genders are displayed on the Rename Tabs and Labels page in Setup but are stored internally as "Feminine". When setting them through the Metadata API, use "Feminine". • `Euter (Swedish)` • `Common (Dutch)`
Picklist (Including Dependent Picklist)	Custom field	(This field type isn't used in Metadata API. CustomField includes this field type for Tooling API support). Represents a picklist, a set of labels and values that can be selected from a picklist.
SharingModel	Custom object	Represents the sharing model for the custom object. Depending on the object, valid values are: • `Private` • `Read` • `ReadWrite` • `ReadWriteTransfer` • `FullAccess` • `ControlledByParent` • `ControlledByCampaign` • `ControlledByLeadOrContact` For example, the User object supports `Private` and `Read` values. Accounts, opportunities, and custom objects support `Private`, `Read` and `ReadWrite` values. Campaign members support `ControlledByCampaign` and `ControlledByLeadOrContact`.
StartsWith	Custom object Custom field	Indicates whether the noun starts with a vowel, consonant, or is a special character. This is used for languages where words need different treatment depending on the first character. Valid values are: • `Consonant` • `Vowel` • `Special (for nouns starting with z, or s plus consonants)`
TreatBlanksAs	Custom field	Indicates how blanks should be treated. Valid values are: • `BlankAsBlank` • `BlankAsZero`
ValueSet	Custom field	Represents a set of values that can be selected from a custom picklist field. Defines the valueSet of a custom picklist field.

ValueSet

Represents a set of values that can be selected from a custom picklist field. Defines the valueSet of a custom picklist field.

Field Type	Field Type	Description
`controllingField`	string	The `fullname` of the controlling field if this is a dependent picklist. A controlling field can be a checkbox or picklist field, but in this case it's a picklist. The controlling picklist filters the available values in the dependent picklist.
`restricted`	boolean	Whether the picklist's values are limited to only the values defined by a Salesforce admin. Values are `true` or `false`.
`valueSetDefinition`	ValueSetValuesDefinition	Defines value-specific settings for a custom dependent picklist. Indicates whether the value set of the custom picklist field is sorted alphabetically.
`valueSetName`	string	The `masterLabel` of the global value set to be used for this picklist field.
`valueSettings`	ValueSettings	Used for the settings that describe a value in a custom picklist field. The picklist can have its own unique value set, or inherit the values from a global value set. You can add field dependency values via Metadata API but not remove them.

ValueSetValuesDefinition

Field Name	Field Type	Description
`sorted`	boolean	Whether the picklist's value set is displayed in alphabetical order in the user interface.
`value`	CustomValue	Required. The list of values for this local, custom picklist.

ValueSettings

Field Name	Field Type	Description
`controllingFieldValue`	stringstring[]	Applies only to dependent custom picklists. A list of values in the controlling or parent picklist (that the custom picklist values depend on).
`valueName`	string	Defines the values in the custom dependent picklist.

CustomObjectTranslation

This metadata type allows you to translate custom objects for a variety of languages.

This type extends the Metadata metadata type and inherits its `fullName` field. The ability to translate component labels is part of the Translation Workbench. For more information, see “Enable and Disable the Translation Workbench” in the Salesforce online help.

Declarative Metadata File Suffix and Directory Location

Translations are stored in a file with a format of *customObjectName__c-lang*.objectTranslation, where *customObjectName__c* is the custom object name, and *lang* is the translation language. A sample file name for German translations is myCustomObject__c-de.objectTranslation.

Custom object translations are stored in the objectTranslations folder in the corresponding package directory.

Version

CustomObjectTranslation components are available in API version 14.0 and later.

Fields

Field	Field Type	Description
caseValues	ObjectNameCaseValue[]	Different combinations of the custom object with regard to article, plural, possessive, and case.
fields	CustomFieldTranslation[]	A list of translations for the custom fields associated with the custom object.
fieldSets	FieldSetTranslation[]	A list of field set translations. Available in API version 41.0 and later.
fullName	string	The name of the custom object and the translation language with a format of *customObjectName-lang*, where *customObjectName* is the custom object name, and *lang* is the translation language. Inherited from Metadata, this field is not defined in the WSDL for this metadata type. It must be specified when creating, updating, or deleting. See createMetadata() to see an example of this field specified for a call.
gender	Gender	Indicates the gender of the noun that represents the object. This is used for languages where words need different treatment depending on their gender.
layouts	LayoutTranslation[]	A list of page layout translations.
nameFieldLabel	string	The label for the name field. Maximum of 765 characters.
namedFilters	NamedFilterTranslation[]	A list of translations for lookup filter error messages associated with the custom object. This field has been removed as of API version 30.0 and is only available in prior versions. The translation metadata associated with a lookup filter is now represented by the lookupFilter field in the CustomFieldTranslation subtype.
quickActions	QuickActionTranslation[]	A list of translations for actions.

Field	Field Type	Description
recordTypes	RecordTypeTranslation[]	A list of record type translations.
sharingReasons	SharingReasonTranslation[]	A list of sharing reason translations.
startsWith	StartsWith (enumeration of type string)	Indicates whether the noun starts with a vowel, consonant, or is a special character. This is used for languages where words need different treatment depending on the first character.
validationRules	ValidationRuleTranslation[]	A list of validation rule translations.
webLinks	WebLinkTranslation[]	A list of web link translations.
workflowTasks	WorkflowTaskTranslation[]	A list of workflow task translations.

CustomFieldTranslation

CustomFieldTranslation contains details for a custom field translation. In API versions 37.0 and earlier standard picklist values could be translated with CustomFieldTranslation. In API version 38.0, use StandardValueSetTranslation instead. For more details, see CustomField.

Note: Not every language supports all the possible values for the fields in CustomFieldTranslation. For language-specific supported values, see the fully supported languages and end-user languages appendices.

Field	Field Type	Description
caseValues	ObjectNameCaseValue[]	Different combinations of the custom object with regard to article, plural, possessive, and case. Available in API version 29.0 and later.
description	string	Translation for the custom field description.
gender	Gender	Indicates the gender of the noun that represents the object. This is used for languages where words need different treatment depending on their gender. Available in API version 29.0 and later.
help	string	Translation for the text that displays in the field-level help hover text for this field.
label	string	Translation for the label. Maximum of 765 characters.
lookupFilter	LookupFilterTranslation	Represents the translation metadata associated with a lookup filter. This field is available in API version 30.0 and later. Note: LookupFilter is not supported on the article type object.
name	string	Required. The name of the field relative to the custom object; for example, `MyField__c`.

Field	Field Type	Description
`picklistValues`	PicklistValueTranslation[]	List of translations for picklist values. See PicklistValue. Note: “Subject” on the Task object is a text field, not a picklist value. It can’t be retrieved via Metadata API. Translations can be provided via the Translation Workbench.
`relationshipLabel`	string	Translation for a lookup relationship label. A lookup relationship allows a field to be associated with another field. The relationship field allows users to select an option from a list of values defined by the other field. Maximum of 765 characters.
`startsWith`	StartsWith (enumeration of type string)	Indicates whether the noun starts with a vowel, consonant, or is a special character. This is used for languages where words need different treatment depending on the first character. Available in API version 29.0 and later.

FieldSetTranslation

FieldSetTranslation contains details for a field set translation. For more details, see FieldSet. Available in API 41.0 and later.

Field	Field Type	Description
`label`	string	Required. Translation for the field set label. Maximum of 765 characters.
`name`	string	Required. The field set name.

LayoutTranslation

LayoutTranslation contains details for a page layout translation. For more details, see Fields.

Field	Field Type	Description
`layout`	string	Required. The layout name.
`layoutType`	string	
`sections`	LayoutSectionTranslation[]	An array of layout section translations.

LayoutSectionTranslation

LayoutSectionTranslation contains details for a page layout section translation. For more details, see LayoutSection.

Field	Field Type	Description
`label`	string	Required. Translation for the label. Maximum of 765 characters.
`section`	string	Required. The section name.

LookupFilterTranslation

LookupFilterTranslation shows a translation for a lookup filter error message associated with the custom object. Replaces NamedFilterTranslation.

LookupFilterTranslation is available in API version 30.0 and later.

Field	Field Type	Description
`errorMessage`	string	The error message that appears if the lookup filter fails.
`informationalMessage`	string	The information message displayed on the page. Use to describe things the user might not understand, such as why certain items are excluded in the lookup filter.

NamedFilterTranslation

NamedFilterTranslation has been removed as of API version 30.0 and is only available in previous API versions.

NamedFilterTranslation shows a list of translations for lookup filter error messages associated with the custom object. See NamedFilter for more information.

Field	Field Type	Description
`errorMessage`	string	The error message that appears if the lookup filter fails.
`informationalMessage`	string	The information message displayed on the page. Use to describe things the user might not understand, such as why certain items are excluded in the lookup filter.
`name`	string	Required. The name of the lookup filter. If you create this field in the user interface, a name is automatically assigned. If you create this field through Metadata API, you must include the `name` field.

ObjectNameCaseValue

ObjectNameCaseValue supports multiple cases and definitions of the custom object name to allow usage in various grammatical contexts.

Note: Not every language supports all the possible values for the fields in ObjectNameCaseValue. For language-specific supported values, see the fully supported languages and end-user languages appendices.

Field	Field Type	Description
`article`	Article (enumeration of type string)	English has two types of articles: definite (*`the`*) and indefinite (*`a`*, *`an`*). The usage of these articles depends mainly on whether you are referring to any member of a group, or to a specific member of a group. The valid values are: • `Definite` • `Indefinite` • `None`

Field	Field Type	Description
caseType	CaseType (enumeration of type string)	The case of the custom object name. The valid values are: - Ablative - Accusative - Adessive - Allative - Causalfinal - Dative - Delative - Distributive - Elative - Essive - Essiveformal - Genitive - Illative - Inessive - Instrumental - Lative - Locative - Nominative - Objective - Partitive - Prepositional - Subjective - Sublative - Superessive - Termanative - Translative - Vocative
plural	boolean	Indicates whether the value field is plural (true) or singular (false).
possessive	Possessive (enumeration of type string)	The possessive case of a language is a grammatical case used to indicate a relationship of possession. The valid values are: - First - None - Second
value	string	Required. The value or label in this grammatical context.

PicklistValueTranslation

PicklistValueTranslation contains details for translation of a picklist value from a local, custom picklist field. For more details, see Picklist (Including Dependent Picklist).

Field	Field Type	Description
`masterLabel`	string	Required. The picklist value defined on the setup page in the application is your master label. The master label is displayed wherever a translated label is not available.
`translation`	string	Required. Translation for the value.

QuickActionTranslation

QuickActionTranslation contains details for an action label in the user interface. For more information, see QuickAction.

Field	Field Type	Description
`label`	string	Required. Translation for the label. Maximum of 765 characters.
`name`	string	Required. The quick action name.

RecordTypeTranslation

RecordTypeTranslation contains details for a record type name translation. For more details, see RecordType.

Field	Field Type	Description
`label`	string	Required. Translation for the label. Maximum of 765 characters.
`name`	string	Required. The record type name.
`description`	string	Translation for the record type description. Available in API version 42.0 and later.

SharingReasonTranslation

SharingReasonTranslation contains details for a sharing reason translation. For more details, see SharingReason.

Field	Field Type	Description
`label`	string	Required. Translation for the sharing reason.
`name`	string	Required. The sharing reason name.

ValidationRuleTranslation

ValidationRuleTranslation contains details for a validation rule translation. For more details, see ValidationRule.

Field	Field Type	Description
errorMessage	string	Required. Translation for the error message associated with the validation rule failure.
name	string	Required. The validation rule name.

WebLinkTranslation

WebLinkTranslation contains details for a web link translation. For more details, see WebLink.

Field	Field Type	Description
label	string	Required. Translation for the web link label. Maximum of 765 characters.
name	string	Required. The web link name.

WorkflowTaskTranslation

WorkflowTaskTranslation contains details for a workflow task translation. For more details, see Workflow.

Field	Field Type	Description
description	string	Translation for the workflow task description.
name	string	Required. The workflow task name.
subject	string	Translation for the workflow task subject.

Declarative Metadata Sample Definitions

This is a sample XML definition of a CustomObjectTranslation for the Description__c object in German, with one custom field, Summary__c. The name and location of the file containing this definition would be objectTranslations/Description__c-de.objectTranslation.

```
<?xml version="1.0" encoding="UTF-8"?>
<CustomObjectTranslation xmlns="http://soap.sforce.com/2006/04/metadata">
    <caseValues>
        <caseType>Nominative</caseType>
        <plural>false</plural>
        <value>Beschreibung</value>
    </caseValues>
    <caseValues>
        <caseType>Nominative</caseType>
        <plural>true</plural>
        <value>Beschreibungen</value>
    </caseValues>
    <caseValues>
        <caseType>Accusative</caseType>
        <plural>false</plural>
        <value>Beschreibung</value>
    </caseValues>
    <caseValues>
        <caseType>Accusative</caseType>
        <plural>true</plural>
        <value>Beschreibungen</value>
    </caseValues>
    <caseValues>
        <caseType>Genitive</caseType>
        <plural>false</plural>
        <value>Beschreibung</value>
    </caseValues>
    <caseValues>
        <caseType>Genitive</caseType>
        <plural>true</plural>
        <value>Beschreibungen</value>
    </caseValues>
    <caseValues>
        <caseType>Dative</caseType>
        <plural>false</plural>
        <value>Beschreibung</value>
    </caseValues>
    <caseValues>
        <caseType>Dative</caseType>
        <plural>true</plural>
        <value>Beschreibungen</value>
    </caseValues>
    <fields>
        <label>Zusammenfassung</label>
        <name>Summary__c</name>
    </fields>
    <gender>Feminine</gender>
    <nameFieldLabel>Beschreibungen</nameFieldLabel>
</CustomObjectTranslation>
```

This is a sample XML definition of a CustomObjectTranslation for the Account object, renaming Account to Client (Kunde) in German. The Account object has one standard field, account_number, and one custom field, Account_Code__c. The name and location of the file containing this definition would be `objectTranslations/Account-de.objectTranslation`.

```
<?xml version="1.0" encoding="UTF-8"?>
<CustomObjectTranslation xmlns="http://soap.sforce.com/2006/04/metadata">
    <caseValues>
        <caseType>Nominative</caseType>
        <plural>false</plural>
        <value>Kunde</value>
    </caseValues>
    <caseValues>
        <caseType>Nominative</caseType>
        <plural>true</plural>
        <value>Kunden</value>
    </caseValues>
    <caseValues>
        <caseType>Accusative</caseType>
        <plural>false</plural>
        <value>Kunden</value>
    </caseValues>
    <caseValues>
        <caseType>Accusative</caseType>
        <plural>true</plural>
        <value>Kunden</value>
    </caseValues>
    <caseValues>
        <caseType>Genitive</caseType>
        <plural>false</plural>
        <value>Kunden</value>
    </caseValues>
    <caseValues>
        <caseType>Genitive</caseType>
        <plural>true</plural>
        <value>Kunden</value>
    </caseValues>
    <caseValues>
        <caseType>Dative</caseType>
        <plural>false</plural>
        <value>Kunden</value>
    </caseValues>
    <caseValues>
        <caseType>Dative</caseType>
        <plural>true</plural>
        <value>Kunden</value>
    </caseValues>
    <fields>
        <caseValues>
            <caseType>Nominative</caseType>
            <plural>false</plural>
            <value>Kundennummer</value>
        </caseValues>
        <caseValues>
            <caseType>Nominative</caseType>
            <plural>true</plural>
            <value>Kundennummern</value>
        </caseValues>
        <gender>Feminine</gender>
        <name>account_number</name>
    </fields>
    <fields>
        <label>Kunden-Code</label>
        <name>Account_Code__c</name>
    </fields>
    <gender>Masculine</gender>
</CustomObjectTranslation>
```

Wildcard Support in the Manifest File

This metadata type supports the wildcard character `*` (asterisk) in the `package.xml` manifest file. For information about using the manifest file, see Deploying and Retrieving Metadata with the Zip File.

SEE ALSO:

CustomObject

Translations

CustomPageWebLink

Represents a custom link defined in a home page component. This type extends the Metadata metadata type and inherits its `fullName` field.

All other custom links are stored as a WebLink in a CustomObject.

Declarative Metadata File Suffix and Directory Location

There is one file per custom link definition, stored in the `weblinks` folder in the corresponding package directory. The file suffix is `.weblink`.

Version

CustomPageWebLinks are available in API version 13.0 and later.

Fields

Field Name	Field Type	Description
`availability`	WebLinkAvailability (enumeration of type string)	Required. Indicates whether the link is only available online (`online`, or if it is also available offline (`offline`).
`description`	string	A description of the link.
`displayType`	WebLinkDisplayType (enumeration of type string)	Represents how this link is rendered. Valid values: • `link` for a hyperlink • `button` for a button • `massActionButton` for a button attached to a related list
`encodingKey`	Encoding (enumeration of type string)	Required. The default encoding setting is Unicode: `UTF-8`. Change it if your template requires data in a different format. This is available if your content source is URL.Valid values include: • `UTF-8`—Unicode (UTF-8) • `ISO-8859-1`—General US & Western Europe (ISO-8859–1, ISO-LATIN-1)

Field Name	Field Type	Description
		• `Shift_JIS`—Japanese (Shift-JIS) • `ISO-2022-JP`—Japanese (JIS) • `EUC-JP`—Japanese (EUC-JP) • `x-SJIS_0213`—Japanese (Shift-JIS_2004) • `ks_c_5601-1987`—Korean (ks_c_5601-1987) • `Big5`—Traditional Chinese (Big5) • `GB2312`—Simplified Chinese (GB2312) • `Big5-HKSCS`—Traditional Chinese Hong Kong (Big5–HKSCS)
`fullName`	string	The name used as a unique identifier for API access. The `fullName` can contain only underscores and alphanumeric characters. It must be unique, begin with a letter, not include spaces, not end with an underscore, and not contain two consecutive underscores.
`hasMenubar`	boolean	If the `openType` is `newWindow`, this field indicates whether to show the browser menu bar for the window (`true` or not (`false`). Otherwise, leave this field empty.
`hasScrollbars`	boolean	If the `openType` is `newWindow`, this field indicates whether to show the scroll bars for the window (`true`) or not (`false`). Otherwise, leave this field empty.
`hasToolbar`	boolean	If the `openType` is `newWindow`, this field indicates whether to show the browser toolbar for the window (`true`) or not (`false`). Otherwise, leave this field empty.
`height`	int	Height in pixels of the window opened by the link. Required if the `openType` is `newWindow`. Otherwise, leave this field empty.
`isResizable`	boolean	If the `openType` is `newWindow`, this field indicates whether to allow resizing of the window (`true`) or not (`false`). Otherwise, leave this field empty.
`linkType`	WebLinkType (enumeration of type string)	Required. Represents whether the content of the button or link is specified by a URL, an sControl, a JavaScript code block, or a Visualforce page. • `url` • `sControl` • `javascript` • `page` • `flow`—Reserved for future use.
`masterLabel`	string	The master label for the link.
`openType`	WebLinkWindowType (enumeration of type string)	Required. When the link is clicked, this field specifies the window style used to display the content. Valid values are: • `newWindow`

Field Name	Field Type	Description
		- `sidebar` - `noSidebar` - `replace` - `onClickJavaScript`
`page`	string	If the value of `linkType` is `page`, this field represents the Visualforce page. Otherwise, leave this field empty.
`position`	WebLinkPosition (enumeration of type string)	If the `openType` is `newWindow`, this field indicates how the new window should be displayed. Otherwise, leave this field empty. Valid values are: - `fullScreen` - `none` - `topLeft`
`protected`	boolean	Required. Indicates whether this component is protected (`true`) or not (`false`). Protected components cannot be linked to or referenced by components created in the installing organization.
`requireRowSelection`	boolean	If the `openType` is `massAction`, this field indicates whether to require individual row selection to execute the action for this button (`true`) or not (`false`). Otherwise, leave this field empty.
`scontrol`	string	If the value of `linkType` is `sControl`, this field represents the name of the sControl. Otherwise, leave this field empty.
`showsLocation`	boolean	If the `openType` is `newWindow`, this field indicates whether or not to show the browser location bar for the window. Otherwise, leave this field empty.
`showsStatus`	boolean	If the `openType` is `newWindow`, this field indicates whether or not to show the browser status bar for the window. Otherwise, leave this field empty.
`url`	string	If the value of `linkType` is `url`, this field represents the URL value. If the value of `linkType` is `javascript`, this field represents the JavaScript content. If the value is neither of these, leave this field empty. Content must be escaped in a manner consistent with XML parsing rules.
`width`	int	Width in pixels of the window opened by the link. Required if the `openType` is `newWindow`. Otherwise, leave this field empty.

Declarative Metadata Sample Definition

The following is the definition of a Weblink. For related samples, see HomePageComponent and HomePageLayout.

```
<?xml version="1.0" encoding="UTF-8"?>
<CustomPageWebLink xmlns="http://soap.sforce.com/2006/04/metadata">
    <availability>online</availability>
    <displayType>button</displayType>
    <encodingKey>UTF-8</encodingKey
    <hasMenubar>false</hasMenubar>
    <hasScrollbars>true</hasScrollbars>
    <hasToolbar>false</hasToolbar>
    <height>600</height>
    <isResizable>true</isResizable>
    <linkType>url</linkType>
    <masterLabel>detailPageButon</masterLabel>
    <openType>newWindow</openType>
    <position>none</position>
    <protected>false</protected>
    <showsLocation>false</showsLocation>
    <showsStatus>false</showsStatus>
    <url>http://google.com</url>
</CustomPageWebLink>
```

Wildcard Support in the Manifest File

This metadata type supports the wildcard character `*` (asterisk) in the `package.xml` manifest file. For information about using the manifest file, see Deploying and Retrieving Metadata with the Zip File.

SEE ALSO:

- HomePageComponent
- HomePageLayout
- WebLink

CustomPermission

Represents a permission that grants access to a custom feature. This type extends the Metadata metadata type and inherits its `fullName` field.

File Suffix and Directory Location

CustomPermission components have the suffix `.customPermission` and are stored in the `customPermissions` folder.

Version

CustomPermission components are available in API version 31.0 and later.

Special Access Rules

As of Spring '20 and later, only users who have one of these permissions can access this object:

- View Setup and Configuration
- Manage Session Permission Set Activations
- Manage Users
- Assign Permission Sets

Fields

Field Name	Field Type	Description
connectedApp	string	The name of the connected app that's associated with this permission. Limit: 80 characters.
description	string	The custom permission description. Limit: 255 characters.
label	string	Required. The custom permission label. Limit: 80 characters.
requiredPermission	CustomPermissionDependencyRequired[]	Indicates which custom permissions are required by the parent custom permission. This field is available in API version 32.0 and later.

CustomPermissionDependencyRequired

CustomPermissionDependencyRequired determines whether a custom permission is required by the parent custom permission. A required custom permission must be enabled when its parent is enabled.

Field Name	Field Type	Description
customPermission	string	Required. The custom permission name.
dependency	boolean	Required. Indicates whether this custom permission is required by the parent custom permission (true) or not (false).

Declarative Metadata Sample Definition

The following is an example of a CustomPermission component.

```
<?xml version="1.0" encoding="UTF-8"?>
<CustomPermission xmlns="http://soap.sforce.com/2006/04/metadata">
   <connectedApp>Acme</connectedApp>
   <description>Read and edit access for Acme accounts.</description>
   <label>Acme Account Full Access</label>
   <requiredPermission>
```

```
      <customPermission>Acme_Account_Read</customPermission>
      <dependency>true</dependency>
   </requiredPermission>
</CustomPermission>
```

The following is an example `package.xml` that references the previous definition, as well as other custom permissions that are associated with a connected app.

```
<?xml version="1.0" encoding="UTF-8"?>
<Package xmlns="http://soap.sforce.com/2006/04/metadata">
   <types>
      <members>Acme</members>
      <name>ConnectedApp</name>
   </types>
   <types>
      <members>Acme_Account_Email_Read</members>
      <members>Acme_Account_Phone_Edit</members>
      <members>Acme_Account_Full_Access</members>
      <members>Acme_Account_Read</members>
      <name>CustomPermission</name>
   </types>
   <types>
      <members>Acme_Account_Email_Read</members>
      <members>Acme_Account_Phone_Edit</members>
      <members>Acme_Account_Full_Access</members>
      <members>Acme_Account_Read</members>
      <name>PermissionSet</name>
   </types>
   <version>48.0</version>
</Package>
```

CustomSite

Represents a Salesforce site. Create public websites and applications that are directly integrated with your Salesforce organization, but don't require users to log in with a username and password.

For more information, see “Salesforce Sites” in the Salesforce online help.

Note: CustomSite does not currently support syndication feeds.

This type extends the Metadata metadata type and inherits its `fullName` field.

Declarative Metadata File Suffix and Directory Location

Lightning Platform CustomSite components are stored in the `sites` directory of the corresponding package directory. The file name matches the site name, and the extension is `.site`.

Version

Lightning Platform CustomSite components are available in API version 14.0 and later.

Fields

Field	Field Type	Description
`active`	boolean	Required. Determines whether the site is active.
`allowHomePage`	boolean	Required. Determines whether the standard home page is visible to public users. This is a new field in API version 15.0.
`allowStandardAnswersPages`	boolean	Determines whether the standard answer pages are visible to public users. This is a new field in API version 19.0.
`allowStandardIdeasPages`	boolean	Required. Determines whether the standard Ideas pages are visible to public users. This is a new field in API version 15.0.
`allowStandardLookups`	boolean	Required. Determines whether the standard lookup pages are visible to public users. This is a new field in API version 15.0.
`allowStandardPortalPages`	boolean	Required. When enabled, authenticated users in this site can access standard Salesforce pages as allowed by their access controls. When disabled, authenticated users in this site can't access standard Salesforce pages, even if their access controls allow it. If your site serves only Visualforce pages, disabling this setting helps add a layer of access protection to your site. This is a new field in API version 39.0.
`allowStandardSearch`	boolean	Required. Determines whether the standard search pages are visible to public users. This is a new field in API version 15.0.
`analyticsTrackingCode`	string	The tracking code associated with your site. Services such as Google Analytics can use this code to track page request data for your site. This field is available in API version 17.0 and later.
`authorizationRequiredPage`	string	The name of the Visualforce page to be displayed when the guest user tries to access a page for which they are not authorized.
`bandwidthExceededPage`	string	The name of the Visualforce page to be displayed when the site has exceeded its bandwidth quota.
`browserXssProtection`	boolean	Required. Determines whether protection against reflected cross-site scripting attacks is enabled. If a reflected cross-site scripting attack is detected, the browser shows a blank page with no content. Available in API version 41.0 and later.

Field	Field Type	Description
`changePasswordPage`	string	The name of the Visualforce page to be displayed when the portal user attempts to change their password for either the portal or for Chatter Answers, when enabled.
`chatterAnswersForgotPasswordConfirmPage`	string	The name of the Visualforce page to be displayed that informs the user that an email has been sent to them with a temporary password. This field is available if Chatter Answers is enabled for your organization. This field is available in API version 27.0 and later.
`chatterAnswersForgotPasswordPage`	string	The name of the Visualforce page to be displayed when a user clicks the link to retrieve a forgotten password. This field is available if Chatter Answers is enabled for your organization. This field is available in API version 27.0 and later.
`chatterAnswersHelpPage`	string	The name of the Visualforce page to be displayed when the user clicks the help link. This field is available if Chatter Answers is enabled for your organization. This field is available in API version 27.0 and later.
`chatterAnswersLoginPage`	string	The name of the Visualforce page to be displayed to allow users to log in to the portal. This field is available if Chatter Answers is enabled for your organization. This field is available in API version 27.0 and later.
`chatterAnswersRegistrationPage`	string	The name of the Visualforce page to be displayed to allow users to register themselves and access the portal. This field is available in API version 27.0 and later.
`clickjackProtectionLevel`	SiteClickjackProtectionLevel (enumeration of type string)	Required. Sets the clickjack protection level. The options are: • `AllowAllFraming` — Allow framing by any page (no protection) • `External` — Allow framing of site or community pages on external domains (good protection) • `SameOriginOnly` — Allow framing by the same origin only (recommended) • `NoFraming` — Don't allow framing by any page (most protection) This field is available in API version 30.0 and later.
`contentSniffingProtection`	boolean	Required. Determines whether the browser is prevented from inferring the MIME type from the document content. If enabled, it also prevents the browser from executing some malicious files

Field	Field Type	Description
		(JavaScript, Stylesheet) as dynamic content. This field is available in API version 41.0 and later.
`cspUpgradeInsecureRequests`	boolean	Required. Determines whether HTTP requests, including third-party domains, are upgraded to HTTPS. This field is available in API version 41.0 and later.
`customWebAddresses`	SiteWebAddress[]	The root custom URLs associated with the site. Saving or deploying a CustomSite replaces all root custom URLs in the site with the root custom URLs in this list. Custom URLs that use a non-root path prefix are not included in this list and are not affected when saving or deploying a CustomSite. This field is available in API version 21.0 and later.
`description`	string	The site description.
`enableAuraRequests`	boolean	Determines whether guest users can view features available only in Lightning (`true`). If set to `false`, Lightning features don't load. This field is available in API version 46.0 and later.
`favoriteIcon`	string	The name of the file to be used for the icon that appears in the browser's address field when visiting the site. Sets the favorite icon for the entire site.
`fileNotFoundPage`	string	The name of the Visualforce page to be displayed when the guest user tries to access a non-existent page.
`forgotPasswordPage`	string	The name of the Visualforce page to be displayed when a user clicks the Forgot Password link on the site's login page. This field is only applicable for Communities sites.
`genericErrorPage`	string	The name of the Visualforce page to be displayed for errors not otherwise specified.
`guestProfile`	string	Read only. The name of the profile associated with the guest user.
`inMaintenancePage`	string	The name of the Visualforce page to be displayed when the site is down for maintenance.
`inactiveIndexPage`	string	The name of the Visualforce page set as the inactive site home page.
`indexPage`	string	Required. The name of the Visualforce page set as the active site home page.
`masterLabel`	string	Required. The name of the site label in the Salesforce user interface.

Field	Field Type	Description
`portal`	string	The name of the portal associated with this site for login access.
`referrerPolicyOriginWhenCrossOrigin`	boolean	Required. Determines whether the referrer header shows only Salesforce.com rather than the entire URL when loading a page. This feature eliminates the potential for a referrer header to reveal sensitive information that could be present in a full URL, such as an org ID. This field is available in API version 41.0 and later.
`requireHttps`	boolean	Required. Determines whether the site requires secure connections (true) or not (false). When false, the site operates normally via insecure connections instead of redirecting to a secure connection.
`requireInsecurePortalAccess`	boolean	Required. Determines whether to override your organization's security settings and exclusively use HTTP when logging in to the associated portal from your site.
`robotsTxtPage`	string	The name of the Visualforce page to be displayed for the `robots.txt` file used by web crawlers.
`serverIsDown`	string	The name of the static resource to be displayed from the cache server when Salesforce servers are down. The static resource must be a public zip file 1 MB or smaller and must contain a page named `maintenance.html` at the root level of the zip file. Other resources in the zip file, such as images or CSS files, can follow any directory structure. This field is available in API version 17.0 and later.
`siteAdmin`	string	The username of the site administrator.
`siteRedirectMappings`	SiteRedirectMapping[]	An array of all URL redirect rules set for your site. This field is available in API version 20.0 and later.
`siteTemplate`	string	The name of the Visualforce page to be used as the site template.
`siteType`	siteType	Required. Identifies whether the site is a Visualforce (Salesforce Sites), Site.com site, or ChatterNetwork (Salesforce Sites).This is a new field in API version 27.0.
`subdomain`	string	Required. Read only. The custom subdomain prefix for the site. For example, if your site URL is `mycompany.force.com/partners`, `mycompany` is the `subdomain`.
`urlPathPrefix`	string	The first part of the path on the site's URL that distinguishes this site from other sites. For example,

Field	Field Type	Description
		if your site URL is `mycompany.force.com/partners`, `partners` is the `urlPathPrefix`.

SiteRedirectMapping

SiteRedirectMapping represents a URL redirect rule on your Salesforce site. For more information, see "Salesforce Sites URL Redirects" in Salesforce Help.

Field	Field Type	Description
`action`	SiteRedirect (enumeration of type string)	Required. The type of the redirect. Available string values are: - `Permanent` - `Temporary`
`isActive`	boolean	The status of the redirect: active or inactive.
`source`	string	Required. The URL that you want to redirect. It must be a relative URL, but can have any valid extension type, such as `.html` or `.php`.
`target`	string	Required. The new URL you want users to visit. It can be a relative URL or a fully-qualified URL with an `http://` or `https://` prefix.

SiteWebAddress

Represents the web address of a Salesforce site.

Field	Field Type	Description
`certificate`	string	Identifies the certificate associated with the custom domain. If the custom domain is set up for Salesforce to serve HTTPS, this field indicates which certificate to use.
`domainName`	string	Required. The domain of the website, in the form of `www.acme.com`.
`primary`	boolean	Required. Indicates whether this is the primary domain (`true`). If `false`, this is not the primary domain.

Declarative Metadata Sample Definition

A sample XML definition of a site is shown below.

```
<?xml version="1.0" encoding="UTF-8"?>
<CustomSite xmlns="http://soap.sforce.com/2006/04/metadata">
    <active>true</active>
    <allowHomePage>true</allowHomePage>
    <allowStandardIdeasPages>true</allowStandardIdeasPages>
    <allowStandardLookups>true</allowStandardLookups>
    <allowStandardPortalPages>true</allowStandardPortalPages>
    <allowStandardSearch>true</allowStandardSearch>
    <authorizationRequiredPage>Unauthorized</authorizationRequiredPage>
    <bandwidthExceededPage>BandwidthExceeded</bandwidthExceededPage>
    <changePasswordPage>ChangePassword</changePasswordPage>

<chatterAnswersForgotPasswordConfirmPage>ChatterAnswersForgotPasswordConfirm</chatterAnswersForgotPasswordConfirmPage>


<chatterAnswersForgotPasswordPage>ChatterAnswersForgotPassword</chatterAnswersForgotPasswordPage>

    <chatterAnswersHelpPage>ChatterAnswersHelp</chatterAnswersHelpPage>
    <chatterAnswersLoginPage>ChatterAnswersLogin</chatterAnswersLoginPage>

<chatterAnswersRegistrationPage>ChatterAnswersRegistration</chatterAnswersRegistrationPage>

    <clickjackProtectionLevel>SameOriginOnly</clickjackProtectionLevel>
    <customWebAddresses>
      <domainName>www.testing123.com</domainName>
      <primary>true</primary>
    </customWebAddresses>
    <favoriteIcon>myFavIcon</favoriteIcon>
    <fileNotFoundPage>FileNotFound</fileNotFoundPage>
    <genericErrorPage>Exception</genericErrorPage>
    <inMaintenancePage>InMaintenance</inMaintenancePage>
    <serverIsDown>MyServerDownResource</serverIsDown>
    <indexPage>UnderConstruction</indexPage>
    <masterLabel>customSite</masterLabel>
    <portal>Customer Portal</portal>
    <requireInsecurePortalAccess>false</requireInsecurePortalAccess>
    <siteAdmin>admin@myco.org</siteAdmin>
    <siteTemplate>SiteTemplate</siteTemplate>
    <subdomain>myco</subdomain>
</CustomSite>
```

Wildcard Support in the Manifest File

This metadata type supports the wildcard character `*` (asterisk) in the `package.xml` manifest file. For information about using the manifest file, see Deploying and Retrieving Metadata with the Zip File.

SEE ALSO:

Portal

CustomTab

Represents a custom tab. Custom tabs let you display custom object data or other web content in Salesforce. When you add a custom tab to an app in Salesforce Classic, it displays as a tab. When you add a custom tab to an app in Lightning Experience, it displays as an item in the app's navigation bar. When a tab displays a custom object, the tab name is the same as the custom object name; for page, s-control, or URL tabs, the name is arbitrary.

For more information, see "Show Custom Object Data for Your Users" in Salesforce Help. This type extends the Metadata metadata type and inherits its `fullName` field.

File Suffix and Directory Location

The file suffix is `.tab`. There is one file for each tab, stored in the `tabs` folder in the corresponding package directory.

Note: Retrieving a component of this metadata type in a project makes the component appear in any Profile and PermissionSet components that are retrieved in the same package.

Version

Tabs are available in API version 10.0 and later.

Fields

This metadata type contains the following fields:

Field Name	Field Type	Description
`actionOverrides`	ActionOverride[]	A list of the action overrides that are assigned to the tab. Only one override is allowed per `formFactor` for a given tab. This field is available in API version 37.0 and later.
`auraComponent`	string	The name of the Aura component to display in this tab. Only one of these fields can have a value set: • `auraComponent` • `customObject` • `flexiPage` • `lwcComponent` • `page` • `scontrol` • `url`
`customObject`	boolean	Indicates whether this tab is for a custom object (`true`) or not (`false`). If set to `true`, the name of the tab matches the name of the custom object. Only one of these fields can have a value set: • `auraComponent`

Field Name	Field Type	Description
		- customObject - flexiPage - lwcComponent - page - scontrol - url
description	string	The optional description text for the tab.
flexiPage	string	The name of the Lightning page to display in this tab. Only one of these fields can have a value set: - auraComponent - customObject - flexiPage - lwcComponent - page - scontrol - url
frameHeight	int	The height, in pixels of the tab frame. Required for s-control and page tabs.
fullName	string	The name of the tab. The value of this field depends on the type of tab, and the API version. - For custom object tabs, the fullName is the developer-assigned name of the custom object (MyCustomObject__c, for example). For custom object tabs, this name must be the same as the custom object name, and customObject should be set to true. - For web tabs, the fullName is the developer-assigned name of the tab (MyWebTab, for example). The fullName can contain only underscores and alphanumeric characters. It must be unique, begin with a letter, not include spaces, not end with an underscore, and not contain two consecutive underscores. This field is inherited from the Metadata component.
hasSidebar	boolean	Indicates if the tab displays the sidebar panel.
icon	string	The optional reference to the image document for the tab if the tab is not using one of the standard tab styles. This is a new field in API version 14.0.
label	string	This is the label of the tab, for web tabs only.
lwcComponent	string	The name of the Lightning web component to display in this tab. Only one of these fields can have a value set:

Field Name	Field Type	Description
		- auraComponent - customObject - flexiPage - lwcComponent - page - scontrol - url
motif	string	Required. The tab style for the color scheme and icon for the custom tab. For example, "'Custom70: Handsaw," is the handsaw icon.
page	string	The name of the Visualforce page to display in this tab. Only one of these fields can have a value set: - auraComponent - customObject - flexiPage - lwcComponent - page - scontrol - url
scontrol	string	The name of the s-control to display in this tab. Only one of these fields can have a value set: - auraComponent - customObject - flexiPage - lwcComponent - page - scontrol - url
splashPageLink	string	The custom link used as the introductory splash page when users click the tab. References a HomePageComponent.
url	string	The URL for the external web-page to embed in this tab. Only one of these fields can have a value set: - auraComponent - customObject - flexiPage - lwcComponent - page

Field Name	Field Type	Description
		• `scontrol` • `url`
`urlEncodingKey`	Encoding (enumeration of type string)	The default encoding setting is Unicode: `UTF-8`. Change it if you are passing information to a URL that requires data in a different format. This option is available when the value `URL` is selected in the tab type.

Declarative Metadata Sample Definition

The following is the definition of a tab:

```
<?xml version="1.0" encoding="UTF-8"?>
<CustomTab xmlns="http://soap.sforce.com/2006/04/metadata">
    <description>Myriad Publishing</description>
    <frameHeight>600</frameHeight>
    <motif>Custom53: Bell</motif>
    <url>http://www.example.com</url>
    <urlEncodingKey>UTF-8</urlEncodingKey>
</CustomTab>
```

Wildcard Support in the Manifest File

This metadata type supports the wildcard character `*` (asterisk) in the `package.xml` manifest file. For information about using the manifest file, see Deploying and Retrieving Metadata with the Zip File.

SEE ALSO:

CustomApplication

CustomValue

Represents the definition of a value used in a global value set or local custom picklist. Custom picklist fields can be local and unique, or can inherit their values from a global picklist (called a *global value set* in API version 38.0). This type extends the Metadata metadata type and inherits its `fullName` field.

To deactivate a global picklist value, you can invoke an `update()` call on GlobalPicklist (API version 37.0) or GlobalValueSet (API version 38.0 and later) with the value omitted, or with the value's `isActive` field set to `false`. Or, you can invoke an `update()` call directly on GlobalPicklistValue (API version 37.0) or CustomValue (API version 38.0 and later) with the `isActive` field set to `false`.

Note: If picklist values are missing from a component definition, they get deactivated when deployed. Deactivation occurs for picklist values of both standard and custom fields.

CustomValue doesn't support file-based operations and only supports CRUD-based calls. CustomValue is retrieved or deployed together with a GlobalValueSet or CustomObject component.

File Suffix and Directory Location

CustomValue components have the suffix `.customValue`. A CustomValue component is returned with either a GlobalValueSet or CustomObject component.

Version

CustomValue components are available in API version 38.0 and later. CustomValue replaces GlobalPicklistValue from API version 37.0.

Fields

Field Name	Field Type	Description
`color`	string	The color assigned to the picklist value when it's used in charts on reports and dashboards. The color is in hexadecimal format; for example, #FF6600. If a color is not specified, it's assigned dynamically upon chart generation.
`default`	boolean	Required. Indicates whether this value is the default selection for the global picklist and the custom picklists that share its picklist value set. This field is set to *`true`* by default.
`description`	string	A picklist value's description. It's useful to include a description for a picklist value so the reason for creating it can be tracked. Limit: 255 characters.
`isActive`	boolean	Indicates whether this value is currently active or inactive. The default value is *`true`*. Users can select only active values from a picklist. An API retrieve operation for global picklist values returns all active and inactive values in the picklist. (Meanwhile, retrieving the values of a non-global, unrestricted picklist returns only the active values.)
`label`	string	The value's display label. If you don't specify the label when creating a value it defaults to the API name. Available in API version 39.0 and later.

StandardValue

This metadata type defines a value in a value set for a standard picklist and specifies whether this value is the default value. This type extends the CustomValue metadata type and inherits all its fields.

When you deploy changes to standard picklist fields, picklist values are added as needed.

Field Name	Field Type	Description
`allowEmail`	boolean	Indicates whether this value lets users email a quote PDF (`true`), or not (`false`). This field is only relevant for the `Status` field in quotes. This field is available in API version 18.0 and later.
`closed`	boolean	Indicates whether this value is associated with a closed status (`true`), or not (`false`). This field is only relevant for the standard `Status`

Field Name	Field Type	Description
		field in cases and tasks. This field is available in API version 16.0 and up to version 36.0. In version 37.0, this field is in GlobalPicklistValue.
`converted`	boolean	Indicates whether this value is associated with a converted status (`true`), or not (`false`). This field is relevant for only the standard `Lead Status` field in leads. Your organization can set its own guidelines for determining when a lead is qualified, but typically, you want to convert a lead as soon as it becomes a real opportunity that you want to forecast. For more information, see "Convert Qualified Leads" in the Salesforce online help. This field is available in API version 16.0 and later.
`cssExposed`	boolean	Indicates whether this value is available in your Self-Service Portal (`true`), or not (`false`). This field is only relevant for the standard `Case Reason` field in cases. Self-Service provides an online support channel for your customers - allowing them to resolve their inquiries without contacting a customer service representative. For more information about Self-Service, see "Setting Up Your Self-Service Portal" in the Salesforce online help. Note: Starting with Spring '12, the Self-Service portal isn't available for new orgs. Existing orgs continue to have access to the Self-Service portal. This field is available in API version 16.0 and later.
`forecastCategory`	ForecastCategories (enumeration of type string)	Indicates whether this value is associated with a forecast category (`true`), or not (`false`). This field is only relevant for the standard `Stage` field in opportunities. - Omitted - Pipeline - BestCase - Forecast - Closed This field is available in API version 16.0 and later.
`highPriority`	boolean	Indicates whether this value is a high priority item (`true`), or not (`false`). This field is only relevant for the standard `Priority` field in tasks. For more information about tasks, see "Start Using Tasks" in the Salesforce online help. This field is available in API version 16.0 and later.
`probability`	int	Indicates whether this value is a probability percentage (`true`), or not (`false`). This field is only relevant for the standard `Stage` field in opportunities. This field is available in API version 16.0 and later.
`reverseRole`	string	A picklist value corresponding to a reverse role name for a partner. If the role is "subcontractor", then the reverse role might be "general contractor". Assigning a partner role to an account in Salesforce creates

Field Name	Field Type	Description
		a reverse partner relationship so that both accounts list the other as a partner. This field is only relevant for partner roles. For more information, see “Partner Fields” in the Salesforce online help. This field is available in API version 18.0 and later.
`reviewed`	boolean	Indicates whether this value is associated with a reviewed status (`true`), or not (`false`). This field is only relevant for the standard `Status` field in solutions. For more information about opportunities, see “Creating Solutions” in the Salesforce online help. This field is available in API version 16.0 and later.
`won`	boolean	Indicates whether this value is associated with a closed or won status (`true`), or not (`false`). This field is only relevant for the standard `Stage` field in opportunities. This field is available in API version 16.0 and later.

Declarative Metadata Sample Definition

For an example of CustomValue components within a GlobalValueSet component that’s referenced by a `package.xml`, see GlobalValueSet.

Dashboard

Represents a dashboard. Dashboards are visual representations of data that allow you to see key metrics and performance at a glance.

This type extends the Metadata metadata type and inherits its `fullName` field. For more information, see “Edit Dashboards in Accessibility Mode in Salesforce Classic” in the Salesforce online help.

Declarative Metadata File Suffix and Directory Location

Dashboards are stored in the `dashboards` directory of the corresponding package directory. The file name matches the dashboard title and the extension is `.dashboard`.

Retrieving Dashboards

You can’t use the wildcard (*) symbol with dashboards in `package.xml`. To retrieve the list of dashboards for populating `package.xml` with explicit names, call `listMetadata()` and pass in `DashboardFolder` as the type. Note that DashboardFolder is not returned as a type in `describeMetadata()`. Dashboard is returned from `describeMetadata()` with an associated attribute of `inFolder` set to true. If that attribute is set to true, you can construct the type by using the component name with the word Folder, such as DashboardFolder.

The following example shows folders in `package.xml`. The names used in `package.xml` must be developer names, not dashboard titles.

```
<?xml version="1.0" encoding="UTF-8"?>
<Package xmlns="http://soap.sforce.com/2006/04/metadata">
    <types>
```

```
        <members>MyDBFolder/MyDBName</members>
        <name>Dashboard</name>
    </types>
    <types>
        <members>MyDocumentFolder/MyDocumentName</members>
        <name>Document</name>
    </types>
    <types>
        <members>unfiled$public/MarketingProductInquiryResponse</members>
        <members>unfiled$public/SalesNewCustomerEmail</members>
        <name>EmailTemplate</name>
    </types>
    <types>
        <members>MyReportFolder/MyReportName</members>
        <name>Report</name>
    </types>
    <version>48.0</version>
</Package>
```

Version

Dashboard components are available in API version 14.0 and later.

Fields

Field	Field Type	Description
backgroundEndColor	string	Required. A dashboard can have a gradient color change on its charts. This field defines the second color for the gradient and backgroundStartColor defines the first color. If you prefer your background to be all one color or do not want a gradient color change, select the same color for this field and backgroundStartColor. The color is in hexadecimal format; for example #FF6600.
backgroundFadeDirection	ChartBackgroundDirection (enumeration of type string)	Required. The direction of the gradient color change, defined by the backgroundStartColor and backgroundEndColor fields. The valid values are: • Diagonal • LeftToRight • TopToBottom
backgroundStartColor	string	Required. The starting color for the gradient color change on the dashboard's charts. See backgroundEndColor for more information. The color is in hexadecimal format; for example #FF6600.

Field	Field Type	Description
chartTheme	ChartTheme (enumeration of type string)	Determines the default theme for all dashboard charts. Replaces dashboardChartTheme for API v42.0 and later. - light—Light-colored theme. - dark—Dark-colored theme. This field is available in API version 42.0 and later.
colorPalette	ChartColorPalettes (enumeration of type string)	Determines the default palette for all dashboard charts. Replaces dashboardColorPalette for API v42.0 and later. - accessible - bluegrass - colorSafe - Default - dusk - earth - fire - gray - heat - justice - nightfall - pond - sunrise - tropic - unity - water - watermelon This field is available in API version 42.0 and later.
dashboardChartTheme	ChartTheme (enumeration of type string)	Determines the default theme for all dashboard charts. - light—Light-colored theme. - dark—Dark-colored theme. This field is available to maintain backward compatibility with versions prior to API version 42.0.
dashboardColorPalette	ChartColorPalettes (enumeration of type string)	Determines the default palette for all dashboard charts. - accessible - bluegrass - colorSafe - Default - dusk

Field	Field Type	Description
		• earth • fire • gray • heat • justice • nightfall • pond • sunrise • tropic • unity • water • watermelon This field is available to maintain backward compatibility with versions prior to API version 42.0.
dashboardFilters	DashboardFilter[]	The list of filters in a dashboard. This field is available in API version 23.0 and later.
dashboardGridLayout	DashboardGridLayout	Lists the included DashboardGridComponent objects, specifies the number of dashboard columns, and sets each dashboard row's height in pixels. This field is available in API version 35.0 and later.
dashboardType	DashboardType (enumeration of type string)	Determines the way visibility settings are set for a dashboard. The valid values are: • SpecifiedUser—All users see data at the access level of one specific running user, specified in the runningUser field, regardless of their own security settings. • LoggedInUser—Each logged-in user sees data according to his or her own access level. • MyTeamUser—Managers can choose to view the dashboard from the point of view of their subordinates in the role hierarchy. This value is available in API version 20.0 and later. This field is available in API version 19.0 and later.
description	string	Description for the dashboard. Maximum of 255 characters.
folderName	string	Name of the folder that houses the dashboard. This field is available in API version 35.0 and later.

Field	Field Type	Description
`fullName`	string	Inherited from Metadata, this field is not defined in the WSDL for this metadata type. It must be specified when creating, updating, or deleting. See `createMetadata()` to see an example of this field specified for a call. This field specifies the folder and dashboard title; for example `folderSales/California`.
`isGridLayout`	boolean	Specifies whether a dashboard uses the Lightning Experience layout (`true`) or not (`false`). Lightning Experience allows dashboards with more than three columns with components that span multiple columns and multiple rows in size. This field is available in API version 35.0 and later.
`dashboardResultRefreshedDate`	string	Required. Date that the dashboard was last refreshed.
`dashboardResultRunningUser`	string	Required. User currently accessing the dashboard.
`leftSection`	DashboardComponentSection	Required. The left section or column of the dashboard.
`middleSection`	DashboardComponentSection	The middle section or column of the dashboard.
`numSubscriptions`	int	Number of subscriptions reported on the dashboard. This field is available in API version 42.0 and later.
`rightSection`	DashboardComponentSection	Required. The right section or column of the dashboard.
`runningUser`	string	The username of the user whose role and sharing settings are used to determine the data shown in the dashboard. When you deploy a dashboard and the value in this field is not defined or does not correspond to a valid user, the field is populated with the username of the user performing the deployment. Regardless of their security settings, all users viewing a dashboard see exactly the same data, because dashboards are always run using the security settings of a particular user. Tip: To avoid inappropriate exposure of sensitive data, save the dashboard to a folder that is visible only to appropriate users.
`textColor`	string	Required. Color of the text on each chart in the dashboard. The color is in hexadecimal format; for example #FF6600.
`title`	string	Required. The dashboard title.
`titleColor`	string	Required. Color of the titles on each dashboard component. The color is in hexadecimal format; for example #FF6600.

Field	Field Type	Description
titleSize	int	Required. Size of characters in title text. For example, a value of 12 indicates 12pt text.

DashboardFilter

DashboardFilter represents a filter in a dashboard.

Field	Field Type	Description
dashboardFilterOptions	DashboardFilterOption[]	The list of items you can select in the **Filter Options** section of the Add Filter dialog.
name	string	Required. The filter label.

DashboardFilterOption

DashboardFilterOption represents a filter option in a dashboard.

Field	Field Type	Description
operator	DashboardFilterOperation (enumeration of type string)	Required. Represents the filter operation for this filter item. Valid values are: • equals • notEqual • lessThan • greaterThan • lessOrEqual • greaterOrEqual • contains • notContain • startsWith • includes • excludes • between Note: The "between" operator takes two operands (for example, "between MinimumValue, MaximumValue"). Note also that the minimum value is inclusive, while the maximum value is exclusive. All other dashboard filter operations take a single operand only. This field is available in API version 24.0 and later.

Field	Field Type	Description
		With API version 23.0, valid values are enumerated in FilterOperation.
`values`	string[]	Required. One or more values in the **Filter Options** area of the Add Filter dialog. This field is available in API version 24.0 and later.

DashboardGridLayout

Lightning Experience features dashboards with more than three columns and components that span multiple columns and multiple rows in size. DashboardGridLayout lists the included dashboard components, specifies the number of dashboard columns, and sets each dashboard row's height in pixels.

Field	Field Type	Description
`dashboardGridComponents`	DashboardGridComponent[]	List of DashboardGridComponent objects in the dashboard.
`numberOfColumns`	int	Required. Total number of columns in the dashboard.
`rowHeight`	int	Required. Height of each row in pixels.

DashboardGridComponent

Lightning Experience features dashboards with more than three columns and components that span multiple columns and multiple rows in size. DashboardGridComponent specifies location and size of a given dashboard component.

Field	Field Type	Description
`colSpan`	int	Required. The width of the dashboard component in columns. For example, if `colSpan` is 5, then the dashboard component spans five columns.
`columnIndex`	int	Required. The left-most column that is occupied by the dashboard component.
`dashboardComponent`	DashboardComponent	Required. The dashboard component that is being sized and placed.
`rowIndex`	int	Required. The top-most row that is occupied by the dashboard component.
`rowSpan`	int	Required. The height of the dashboard component in rows.

DashboardComponent

A dashboard consists of a group of different components or elements that display data. Each component can use a custom report or a custom s-control as their data source to display corporate metrics or key performance indicators. You can create several dashboard components and display them all in one dashboard aligned in up to three columns.

Field	Field Type	Description
`chartAxisRange`	ChartRangeType (enumeration of type string)	A manual or automatic axis range for bar or line charts. The valid values are: - `auto` - `manual`
`chartAxisRangeMax`	double	The maximum axis range to be displayed. This only applies to bar and line charts in which the `manual` axis range is selected for the `chartAxisRange` field.
`chartAxisRangeMin`	double	The minimum axis range to be displayed. This only applies to bar and line charts in which the `manual` axis range is selected for the `chartAxisRange` field.
`chartSummary`	ChartSummaryc	Specifies the summary field for the chart data. Required if `isAutoSelectFromReport` is set to `false`. This field is available in API version 25.0 and later.
`componentType`	DashboardComponentType (enumeration of type string)	Required. Dashboard component type. The valid values are: - `Bar` - `BarGrouped` - `BarStacked` - `BarStacked100` - `Column` - `ColumnGrouped` - `ColumnLine` - `ColumnLineGrouped` - `ColumnLineStacked` - `ColumnLineStacked100` - `ColumnStacked` - `ColumnStacked100` - `Donut` - `FlexTable` - `Funnel` - `Gauge` - `Line` - `lineCumulative`

Field	Field Type	Description
		- `LineGrouped` - `lineGroupedCumulative` - `Metric` - `Pie` - `Scatter` - `ScatterGrouped` - `Scontrol` - `Table`
`dashboardFilterColumns`	DashboardFilterColumn[]	A list of dashboard filter columns. Each report-based component must have a dashboard filter column that defines the column that the filter applies to. This field is available in API version 23.0 and later.
`dashboardTableColumn`	DashboardTableColumn[]	Represents a list of columns on a customized dashboard table component.
`displayUnits`	ChartUnits (enumeration of type string)	Chart Units. The valid values are: - `Auto` - `Integer` - `Hundreds` - `Thousands` - `Millions` - `Billions` - `Trillions`
`drillDownUrl`	string	For charts, specifies a URL that users go to when they click the dashboard component. Use this option to send users to another dashboard, report, record detail page, or other system that uses a Web interface. This field overrides the `drillEnabled` and `drillToDetailEnabled` fields.
`drillEnabled`	boolean	Specifies whether to take users to the full or filtered source report when they click the dashboard component. Set to `false` to drill to the full source report; set to `true` to drill to the source report filtered by what they clicked. If set to `true`, users can click individual groups, axis values, or legend entries. This overrides the `drillToDetailEnabled` field. This field is available in API version 17.0 and later.
`drillToDetailEnabled`	boolean	When enabled, users are taken to the record detail page when they click a record name, record owner, or feed post

Field	Field Type	Description
		in a table or chart. When set to `true` users can click axis and legend values, chart elements, and table entries. The `drillDownUrl` and `drillEnabled` fields override this field. This field is available in API version 20.0 and later.
`enableHover`	boolean	Specifies whether to display values, labels, and percentages when hovering over charts. Hover details depend on chart type. Percentages apply to pie, donut, and funnel charts only. This field is available in API version 17.0 and later.
`expandOthers`	boolean	Specifies whether to combine all groups less than or equal to 3% of the total into a single 'Others' wedge or segment. This only applies to pie, donut, and funnel charts. Set to `true` to show all values individually on the chart; set to `false` to combine small groups into 'Others.' This field is available in API version 17.0 and later.
`flexComponentProperties`	DashboardFlexTableComponentProperties	Defines metadata for Lightning Experience table columns and sorting. This field is available in API version 41.0 and later.
`footer`	string	Footer displayed at the bottom of the dashboard component. Maximum of 255 characters.
`gaugeMax`	double	The maximum value on a gauge. A gauge is used to see how far you are from reaching a goal. It looks like a speedometer in a car.
`gaugeMin`	double	The minimum value on a gauge.
`groupingColumn`	string	Specifies the field by which to group data. This data is displayed on the X-axis for vertical column charts and on the Y-axis for horizontal bar charts. This field is available in API version 25.0 and later.
`GroupingSortProperties`	DashboardComponentGroupingSortProperties	This field captures sort properties of the dashboard component. If the component has one or more groupings, sort information is stored here; otherwise, it is stored in the `sortBy` field. This field is available in API version 46.0 and later.
`header`	string	Header displayed at the top of the dashboard component. Maximum of 80 characters.
`indicatorBreakpoint1`	double	The value that separates the `indicatorLowColor` from the `indicatorMiddleColor` on the dashboard.

Field	Field Type	Description
indicatorBreakpoint2	double	The value that separates the indicatorMiddleColor from the indicatorHighColor on the dashboard.
indicatorHighColor	string	The color representing a high number range on the gauge.
indicatorLowColor	string	The color representing a low number range on the gauge.
indicatorMiddleColor	string	The color representing a medium number range on the gauge.
legendPosition	ChartLegendPosition (enumeration of type string)	The location of the legend with respect to the chart. The valid values are: • Bottom • OnChart • Right
maxValuesDisplayed	int	The maximum number of elements to include in the top-level grouping of the horizontal axis of a horizontal chart, vertical axis of a vertical chart, or selected axis of a stacked bar chart. For example, if you want to list only your top five salespeople, create an opportunity report that lists total opportunity amounts by owner and enter 5 in this field.
metricLabel	string	Descriptive label for the metric. This is relevant if metric is the value of the componentType field.
page	string	Visualforce page associated with the component.
pageHeightInPixels	int	Display height of the Visualforce page in pixels.
report	string	Name of the report associated with the component.
scontrol	string	S-control associated with component if scontrol is the value of the componentType field. For more information, see "Defining Custom S-Controls" in the Salesforce online help.
scontrolHeightInPixels	int	Display height of the s-control in pixels.
showPercentage	boolean	Indicates if percentages are displayed for regions of gauges and wedges and segments of pie, donut, and funnel charts (true), or not (false).
showPicturesOnCharts	boolean	Display Chatter photos for up to 20 records in a horizontal bar chart component whose source report is grouped by a user or group name field. If there are more than 20 records with photos, record names are shown instead of photos. Set Grouping Display to *None* to show

Field	Field Type	Description
		photos. Set the `Drill Down to` option to *`Record Detail Page`* to take users directly to user profile or group pages when they click photos. Chatter must be enabled for photos to be displayed. Depending on your organization's setup, you may not see photos on tables and charts.
`showPicturesOnTables`	boolean	Display Chatter photos for up to 20 records in a horizontal bar chart component whose source report is grouped by a user or group name field. If there are more than 20 records with photos, record names are shown instead of photos. Set `Grouping Display` to *`None`* to show photos. Set the `Drill Down to` option to *`Record Detail Page`* to take users directly to user profile or group pages when they click photos. Chatter must be enabled for photos to be displayed. Depending on your organization's setup, you may not see photos on tables and charts.
`showTotal`	boolean	Indicates if the total of all wedges is displayed for gauges and donut charts (`true`), or not (`false`).
`showValues`	boolean	Indicates if the values of individual records or groups are displayed for charts (`true`), or not (`false`).
`sortBy`	DashboardComponentFilter (enumeration of type string)	The sort option for the dashboard component.
`title`	string	The title of the dashboard component. Maximum of 40 characters.
`useReportChart`	boolean	Specifies whether to use the chart defined in the source report on this dashboard component. The chart settings in the source report determine how the chart displays in the dashboard, and any chart settings you define for the dashboard are overridden. If you defined a combination chart in the source report, use this option to use that combination chart on this dashboard.

DashboardFilterColumn

DashboardFilterColumn represents a filter column in a dashboard.

Field	Field Type	Description
`column`	string	Required. The report column code for the filter.

DashboardTableColumn

DashboardTableColumn represents a column in a customized table component in a dashboard.

Field	Field Type	Description
`aggregateType`	ReportSummaryType[] (enumeration of type string)	Specifies the aggregation type for the table column.
`column`	string	Required. The label of the column to use in the table.
`showTotal`	boolean	Displays the totals for each summarizable column in the dashboard table. This field is available in API version 19.0 and later.
`sortBy`	DashboardComponentFilter (enumeration of type string)	The sort option for the dashboard table component. Sort on just one column per table.

DashboardFlexTableComponentProperties

DashboardFlexTableComponentProperties represents a column in a customized table component in a dashboard.

Field	Field Type	Description
`flexTableColumn`	DashboardComponentColumn	Represents a column in a Lightning Experience table component. This field is available in API version 41.0 and later.
`flexTableSortInfo`	DashboardComponentSortInfo	Represents sorting column and order in a Lightning Experience table component. This field is available in API version 41.0 and later.
`hideChatterPhotos`	boolean	If `true`, hides any photos from Chatter feeds. This field is available in API version 41.0 and later.
`decimalPrecision`	integer	For columns with numeric values, indicates the number of significant digits.

DashboardComponentGroupingSortProperties

DashboardComponentGroupingSortProperties is composed of multiple elements of the type DashboardComponentGroupingSort.

Field	Field Type	Description
`groupingSorts`	DashboardComponentGroupingSort	This field stores sort information for a dashboard at each grouping level of granularity. This field is available in API version 46.0 and later.

DashboardComponentGroupingSort

DashboardComponentGroupingSort specifies properties for sorting on a dashboard component group.

Field	Field Type	Description
groupingLevel	String	Grouping at which this sort configuration is applied.
inheritedReportGroupingSort	String	true if the sort order is picked up from an underlying report for this grouping level.
sortColumn	String	If grouping is sorted by an aggregate, this value is the aggregate value (such as sortColumn). If the grouping is sorted by its own value, this field is null.
sortOrder	String	Ascending or Descending to reflect the sort order.

DashboardComponentColumn

DashboardComponentColumn represents a component column in a dashboard. Available in API version 41.0 and later.

Field	Field Type	Description
breakPoint1	double	The value that separates the lowRangeColor from the midRangeColor on the dashboard.
breakPoint2	double	The value that separates the midRangeColor from the highRangeColor on the dashboard.
breakPointOrder	double	Conditional highlighting can be applied to multiple columns. This field stores the order of conditional highlights.
highRangeColor	int	The color representing a high number range on the column.
lowRangeColor	int	The color representing a low number range on the column.
midRangeColor	int	The color representing a mid number range on the column.
reportColumn	string	Required. The report column code for the filter.
showTotal	boolean	If true, the column total is displayed.
type	DashboardComponentColumnType (enumeration of type string)	Represents the type of Lightning Experience table column: • Details • Aggregates • Grouping This field is available in API version 41.0 and later.

DashboardComponentSortInfo

DashboardFilterColumns represents a filter column in a dashboard.

Field	Field Type	Description
ComponentSortColumn	string	Indicates the column on which the table is sorted. This field is available in API version 41.0 and later.
sortOrder	string	Indicates whether column sorting is ascending or descending. This field is available in API version 41.0 and later.

DashboardComponentSection

DashboardComponentSection represents one of the sections or columns in a dashboard.

Field	Field Type	Description
columnSize	DashboardComponentSize (enumeration of type string)	Required. The size of the column in the dashboard: • Medium • Narrow • Wide
components	DashboardComponent[]	The list of DashboardComponent objects in the dashboard column.

DashboardComponentFilter

DashboardComponentFilter is an enumeration of type string that lists the sort values for dashboard components. The valid values are:

Enumeration Value	Description
RowLabelAscending	Sorts in alphabetical order by the label.
RowLabelDescending	Sorts in reverse alphabetical order by the label.
RowValueAscending	Sorts lowest to highest by the value.
RowValueDescending	Sorts highest to lowest by the value.

Declarative Metadata Sample Definition — Filtered Dashboard

A sample XML definition of a filtered dashboard is shown below. Note that this example is supported in API version 24.0 and later. The file name matches the dashboard title and the extension is `.dashboard`.

```
<?xml version="1.0" encoding="UTF-8"?>
<Dashboard xmlns="http://soap.sforce.com/2006/04/metadata">
    <backgroundEndColor>#FFFFFF</backgroundEndColor>
    <backgroundFadeDirection>Diagonal</backgroundFadeDirection>
    <backgroundStartColor>#FFFFFF</backgroundStartColor>
    <dashboardFilters>
        <dashboardFilterOptions>
            <operator>equals</operator>
            <values>Media</values>
        </dashboardFilterOptions>
        <dashboardFilterOptions>
            <operator>lessThan</operator>
            <values>Working</values>
        </dashboardFilterOptions>
        <dashboardFilterOptions>
            <operator>between</operator>
            <values>ABC</values>
            <values>XYZ</values>
        </dashboardFilterOptions>
        <name>Industry</name>
    </dashboardFilters>
    <dashboardFilters>
        <dashboardFilterOptions>
            <operator>equals</operator>
            <values>Analyst,Partner</values>
        </dashboardFilterOptions>
        <dashboardFilterOptions>
            <operator>startsWith</operator>
            <values>Integrator</values>
        </dashboardFilterOptions>
        <name>Account Type</name>
    </dashboardFilters>
    <dashboardType>SpecifiedUser</dashboardType>
    <leftSection>
        <columnSize>Medium</columnSize>
        <components>
            <chartAxisRange>Auto</chartAxisRange>
            <componentType>Bar</componentType>
            <dashboardFilterColumns>
                <column>INDUSTRY</column>
            </dashboardFilterColumns>
            <dashboardFilterColumns>
                <column>TYPE</column>
            </dashboardFilterColumns>
            <displayUnits>Auto</displayUnits>
            <drillEnabled>false</drillEnabled>
            <drillToDetailEnabled>false</drillToDetailEnabled>
            <enableHover>false</enableHover>
            <expandOthers>false</expandOthers>
            <legendPosition>Bottom</legendPosition>
            <report>unfiled$public/SampleReportofAccounts</report>
            <showPercentage>false</showPercentage>
            <showPicturesOnCharts>false</showPicturesOnCharts>
            <showValues>false</showValues>
            <sortBy>RowLabelAscending</sortBy>
            <useReportChart>false</useReportChart>
        </components>
    </leftSection>
    <middleSection>
        <columnSize>Medium</columnSize>
        <components>
            <chartAxisRange>Auto</chartAxisRange>
            <componentType>Funnel</componentType>
            <dashboardFilterColumns>
                <column>ACCOUNT_INDUSTRY</column>
            </dashboardFilterColumns>
            <dashboardFilterColumns>
                <column>ACCOUNT.TYPE</column>
            </dashboardFilterColumns>
            <displayUnits>Auto</displayUnits>
            <drillEnabled>false</drillEnabled>
            <drillToDetailEnabled>false</drillToDetailEnabled>
            <enableHover>false</enableHover>
            <expandOthers>false</expandOthers>
            <legendPosition>Bottom</legendPosition>
            <report>unfiled$public/SampleReportofCases</report>
            <showPercentage>false</showPercentage>
            <showValues>true</showValues>
            <sortBy>RowLabelAscending</sortBy>
            <useReportChart>false</useReportChart>
        </components>
    </middleSection>
    <rightSection>
        <columnSize>Medium</columnSize>
        <components>
            <chartAxisRange>Auto</chartAxisRange>
            <componentType>Column</componentType>
            <dashboardFilterColumns>
                <column>INDUSTRY</column>
            </dashboardFilterColumns>
            <dashboardFilterColumns>
                <column>ACCOUNT_TYPE</column>
            </dashboardFilterColumns>
            <displayUnits>Auto</displayUnits>
            <drillEnabled>false</drillEnabled>
            <drillToDetailEnabled>false</drillToDetailEnabled>
            <enableHover>false</enableHover>
            <expandOthers>false</expandOthers>
            <legendPosition>Bottom</legendPosition>
            <report>unfiled$public/SampleReportofOpportunities</report>
            <showPercentage>false</showPercentage>
            <showValues>false</showValues>
            <sortBy>RowLabelAscending</sortBy>
            <useReportChart>false</useReportChart>
        </components>
    </rightSection>
    <runningUser>admin@TESTORGNUM</runningUser>
    <textColor>#000000</textColor>
    <title>My Dashboard</title>
    <titleColor>#000000</titleColor>
    <titleSize>12</titleSize>
</Dashboard>
```

Declarative Metadata Sample Definition — Unfiltered Dashboard

A sample XML definition of a dashboard is shown below. The file name matches the dashboard title and the extension is `.dashboard`.

```
<?xml version="1.0" encoding="UTF-8"?>
<Dashboard xmlns="http://soap.sforce.com/2006/04/metadata">
    <backgroundEndColor>#FFFFFF</backgroundEndColor>
    <backgroundFadeDirection>LeftToRight</backgroundFadeDirection>
    <backgroundStartColor>#FFFFFF</backgroundStartColor>
    <description>Dashboard with all possible chart types</description>
    <leftSection>
        <columnSize>Medium</columnSize>
        <components>
            <chartAxisRange>Auto</chartAxisRange>
            <componentType>BarStacked100</componentType>
            <displayUnits>Auto</displayUnits>
            <drillEnabled>true</drillEnabled>
            <enableHover>true</enableHover>
            <report>testFolder/sourceRep</report>
            <sortBy>RowLabelAscending</sortBy>
        </components>
        <components>
            <componentType>Table</componentType>
            <dashboardTableColumn>
                <column>CLOSE_DATE</column>
                <sortBy>RowLabelAscending</sortBy>
            </dashboardTableColumn>
            <dashboardTableColumn>
                <aggregateType>Sum</aggregateType>
                <column>AMOUNT</column>
                <showTotal>true</showTotal>
            </dashboardTableColumn>
            <dashboardTableColumn>
                <column>STAGE_NAME</column>
            </dashboardTableColumn>
            <dashboardTableColumn>
                <column>PROBABILITY</column>
                <aggregateType>Maximum</aggregateType>
            </dashboardTableColumn>
            <displayUnits>Integer</displayUnits>
            <header>Opportunities Table</header>
            <indicatorHighColor>#54C254</indicatorHighColor>
            <indicatorLowColor>#C25454</indicatorLowColor>
            <indicatorMiddleColor>#C2C254</indicatorMiddleColor>
            <maxValuesDisplayed>10</maxValuesDisplayed>
            <report>testFolder/sourceRep</report>
        </components>
        <components>
            <chartAxisRange>Auto</chartAxisRange>
            <componentType>Bar</componentType>
            <displayUnits>Auto</displayUnits>
            <drillEnabled>true</drillEnabled>
            <enableHover>true</enableHover>
            <report>testFolder/sourceRep</report>
            <sortBy>RowLabelAscending</sortBy>
        </components>
        <components>
            <chartAxisRange>Auto</chartAxisRange>
            <componentType>Column</componentType>
            <displayUnits>Auto</displayUnits>
            <drillEnabled>true</drillEnabled>
            <legendPosition>Bottom</legendPosition>
            <report>testFolder/sourceRep</report>
            <sortBy>RowLabelAscending</sortBy>
            <useReportChart>true</useReportChart>
        </components>
        <components>
            <chartAxisRange>Auto</chartAxisRange>
            <componentType>Funnel</componentType>
            <displayUnits>Auto</displayUnits>
            <drillEnabled>true</drillEnabled>
            <enableHover>true</enableHover>
            <expandOthers>true</expandOthers>
            <legendPosition>Bottom</legendPosition>
            <report>testFolder/sourceRep</report>
            <sortBy>RowLabelAscending</sortBy>
        </components>
    </leftSection>
    <middleSection>
        <columnSize>Medium</columnSize>
        <components>
            <chartAxisRange>Auto</chartAxisRange>
            <componentType>ColumnStacked100</componentType>
            <displayUnits>Auto</displayUnits>
            <drillEnabled>true</drillEnabled>
            <enableHover>true</enableHover>
            <report>testFolder/sourceRep</report>
            <sortBy>RowLabelAscending</sortBy>
        </components>
        <components>
            <chartAxisRange>Auto</chartAxisRange>
            <componentType>ColumnStacked</componentType>
            <displayUnits>Auto</displayUnits>
            <drillEnabled>true</drillEnabled>
            <enableHover>true</enableHover>
            <report>testFolder/sourceRep</report>
            <sortBy>RowLabelAscending</sortBy>
        </components>
        <components>
            <chartAxisRange>Auto</chartAxisRange>
            <componentType>ColumnStacked</componentType>
            <displayUnits>Auto</displayUnits>
            <drillEnabled>true</drillEnabled>
            <enableHover>true</enableHover>
            <report>testFolder/sourceRep</report>
            <sortBy>RowLabelAscending</sortBy>
        </components>
        <components>
            <chartAxisRange>Auto</chartAxisRange>
            <componentType>ColumnGrouped</componentType>
            <displayUnits>Auto</displayUnits>
            <drillEnabled>true</drillEnabled>
            <enableHover>true</enableHover>
            <report>testFolder/sourceRep</report>
            <sortBy>RowLabelAscending</sortBy>
        </components>
        <components>
            <chartAxisRange>Auto</chartAxisRange>
            <componentType>Column</componentType>
            <displayUnits>Auto</displayUnits>
            <drillEnabled>true</drillEnabled>
            <enableHover>true</enableHover>
            <report>testFolder/sourceRep</report>
            <sortBy>RowLabelAscending</sortBy>
        </components>
    </middleSection>
    <rightSection>
        <columnSize>Medium</columnSize>
        <components>
            <chartAxisRange>Auto</chartAxisRange>
            <componentType>Bar</componentType>
            <displayUnits>Auto</displayUnits>
            <drillEnabled>true</drillEnabled>
            <enableHover>true</enableHover>
            <report>testFolder/sourceRep</report>
            <sortBy>RowLabelAscending</sortBy>
        </components>
        <components>
            <chartAxisRange>Auto</chartAxisRange>
            <componentType>Pie</componentType>
            <displayUnits>Auto</displayUnits>
            <drillEnabled>true</drillEnabled>
            <enableHover>true</enableHover>
            <expandOthers>true</expandOthers>
            <report>testFolder/sourceRep</report>
            <sortBy>RowLabelAscending</sortBy>
        </components>
        <components>
            <chartAxisRange>Auto</chartAxisRange>
            <componentType>LineGroupedCumulative</componentType>
            <displayUnits>Auto</displayUnits>
            <drillEnabled>true</drillEnabled>
            <enableHover>true</enableHover>
            <report>testFolder/sourceRep</report>
            <sortBy>RowLabelAscending</sortBy>
        </components>
        <components>
            <chartAxisRange>Auto</chartAxisRange>
            <componentType>LineGrouped</componentType>
            <displayUnits>Auto</displayUnits>
            <drillEnabled>true</drillEnabled>
            <enableHover>true</enableHover>
            <report>testFolder/sourceRep</report>
            <sortBy>RowLabelAscending</sortBy>
        </components>
        <components>
            <chartAxisRange>Auto</chartAxisRange>
            <componentType>LineCumulative</componentType>
            <displayUnits>Auto</displayUnits>
            <drillEnabled>true</drillEnabled>
            <enableHover>true</enableHover>
            <report>testFolder/sourceRep</report>
            <sortBy>RowLabelAscending</sortBy>
        </components>
        <components>
            <chartAxisRange>Auto</chartAxisRange>
            <componentType>Donut</componentType>
            <displayUnits>Auto</displayUnits>
            <drillEnabled>true</drillEnabled>
            <enableHover>true</enableHover>
            <expandOthers>true</expandOthers>
            <report>testFolder/sourceRep</report>
            <sortBy>RowLabelAscending</sortBy>
        </components>
    </rightSection>
    <runningUser>admin@TESTORGNUM</runningUser>
    <textColor>#000000</textColor>
    <title>Db Title</title>
    <titleColor>#000000</titleColor>
    <titleSize>12</titleSize>
</Dashboard>
```

Declarative Metadata Sample Definition — Lightning Experience Dashboard with `isGridLayout` Equals `true`

A sample XML definition of a Lightning Experience dashboard with `isGridLayout` equals `true` is shown below. Note that this example is supported in API version 35.0 and later. The file name matches the dashboard title and the extension is `.dashboard`.

```
<?xml version="1.0" encoding="UTF-8"?>
<Dashboard xmlns="http://soap.sforce.com/2006/04/metadata">
    <backgroundEndColor>#FFFFFF</backgroundEndColor>
    <backgroundFadeDirection>Diagonal</backgroundFadeDirection>
    <backgroundStartColor>#FFFFFF</backgroundStartColor>
    <dashboardType>SpecifiedUser</dashboardType>
    <gridLayout>
        <dashboardGridComponents>
            <colSpan>3</colSpan>
            <columnIndex>0</columnIndex>
            <dashboardComponent>
                <autoselectColumnsFromReport>false</autoselectColumnsFromReport>
                <chartAxisRange>Auto</chartAxisRange>
                <chartSummary>
                    <axisBinding>y</axisBinding>
                    <column>RowCount</column>
                </chartSummary>
                <componentType>Donut</componentType>
                <drillEnabled>false</drillEnabled>
                <drillToDetailEnabled>false</drillToDetailEnabled>
                <enableHover>false</enableHover>
                <expandOthers>false</expandOthers>
                <groupingColumn>TITLE</groupingColumn>
                <legendPosition>Bottom</legendPosition>
                <report>unfiled$public/lead_rpt</report>
                <showPercentage>false</showPercentage>
                <showTotal>false</showTotal>
                <showValues>true</showValues>
                <sortBy>RowLabelAscending</sortBy>
                <useReportChart>false</useReportChart>
            </dashboardComponent>
            <rowIndex>0</rowIndex>
            <rowSpan>3</rowSpan>
        </dashboardGridComponents>
        <dashboardGridComponents>
            <colSpan>3</colSpan>
            <columnIndex>0</columnIndex>
            <dashboardComponent>
                <autoselectColumnsFromReport>false</autoselectColumnsFromReport>
                <chartAxisRange>Auto</chartAxisRange>
                <chartSummary>
                    <axisBinding>y</axisBinding>
                    <column>RowCount</column>
                </chartSummary>
                <componentType>Pie</componentType>
                <drillEnabled>false</drillEnabled>
                <drillToDetailEnabled>false</drillToDetailEnabled>
                <enableHover>false</enableHover>
                <expandOthers>false</expandOthers>
                <groupingColumn>TITLE</groupingColumn>
                <legendPosition>Bottom</legendPosition>
                <report>unfiled$public/lead_rpt</report>
                <showPercentage>false</showPercentage>
                <showValues>true</showValues>
                <sortBy>RowLabelAscending</sortBy>
                <useReportChart>false</useReportChart>
            </dashboardComponent>
            <rowIndex>3</rowIndex>
            <rowSpan>3</rowSpan>
        </dashboardGridComponents>
        <dashboardGridComponents>
            <colSpan>3</colSpan>
            <columnIndex>0</columnIndex>
            <dashboardComponent>
                <autoselectColumnsFromReport>false</autoselectColumnsFromReport>
                <chartAxisRange>Auto</chartAxisRange>
                <chartSummary>
                    <axisBinding>y</axisBinding>
                    <column>RowCount</column>
                </chartSummary>
                <componentType>Column</componentType>
                <drillEnabled>false</drillEnabled>
                <drillToDetailEnabled>false</drillToDetailEnabled>
                <enableHover>false</enableHover>
                <expandOthers>false</expandOthers>
                <groupingColumn>TITLE</groupingColumn>
                <legendPosition>Bottom</legendPosition>
                <report>unfiled$public/lead_rpt</report>
                <showPercentage>false</showPercentage>
                <showValues>false</showValues>
                <sortBy>RowLabelAscending</sortBy>
                <useReportChart>false</useReportChart>
            </dashboardComponent>
            <rowIndex>9</rowIndex>
            <rowSpan>3</rowSpan>
        </dashboardGridComponents>
        <numberOfColumns>9</numberOfColumns>
        <rowHeight>90</rowHeight>
    </gridLayout>
    <isGridLayout>true</isGridLayout>
    <runningUser>admin@s1.com</runningUser>
    <textColor>#000000</textColor>
    <title>sfx</title>
    <titleColor>#000000</titleColor>
    <titleSize>12</titleSize>
</Dashboard>
```

Wildcard Support in the Manifest File

This metadata type doesn't support the wildcard character `*` (asterisk) in the `package.xml` manifest file. For information about using the manifest file, see Deploying and Retrieving Metadata with the Zip File.

SEE ALSO:

Folder

Report

DataCategoryGroup

Represents a data category group.

This type extends the Metadata metadata type and inherits its `fullName` field.

Warning: Using Metadata API to deploy category changes from one organization to another permanently removes categories and record categorizations that are not specified in your XML file. Salesforce recommends that you manually create data categories and record associations in an organization from Setup by entering *`Data Categories`* in the `Quick Find` box, then selecting **Data Categories** rather than deploying changes from a sandbox to a production organization. For more information, see Usage.

Data category groups are provided to:

- Classify and filter data.
- Share data among users.

Every data category group contains items or data categories that can be organized hierarchically.

The example below shows the `Geography` data category group and its data categories.

```
Geography
    Worldwide
        North America
            United States of America
            Canada
            Mexico
        Europe
        Asia
```

Note: See "Work with Data Categories" in the Salesforce online help for more information on data category groups, data categories, parent and sub categories.

File Suffix and Directory Location

The file suffix is `.datacategorygroup`. There is one file for each data category group stored in the `datacategorygroups` folder in the corresponding package directory.

Version

Data category groups are available in API version 18.0 and later.

Fields

This metadata type contains the following fields:

Field Name	Field Type	Description
`active`	boolean	Required. The status of the category group. Indicates whether this category group is active, (`true`), or not active (`false`).
`dataCategory`	DataCategory	Required. The top-level category within the data category group.
`description`	string	The description of the data category group.
`fullName`	string	Required. The unique name of the data category group. When creating a data category group, the `fullName` field and the file name (without its suffix) must match.The `fullName` can contain only underscores and alphanumeric characters. It must be unique, begin with a letter, not include spaces, not end with an underscore, and not contain two consecutive underscores. This field is inherited from the Metadata component.
`label`	string	Required. Label that represents the object in Salesforce.
`objectUsage`	ObjectUsage	The objects that are associated with the data category group.

DataCategory

Represents an item (or data category) in the data category group. A data category can recursively contain a list of other data categories.

Field Name	Field Type	Description
`dataCategory`	DataCategory[]	A recursive list of sub data categories. For example, a list of countries within a continent. You can create up to 100 categories in a data category group and have up to 5 levels in a data category group hierarchy.
`label`	string	Required. Label for the data category throughout the Salesforce user interface.
`name`	string	Required. The developer name of the data category used as a unique identifier for API access. The name can only contain characters, letters, and the underscore (_) character, must start with a letter, and cannot end with an underscore or contain two consecutive underscore characters. Important: The value for this field is defined once and cannot be changed later. Warning: If you deploy a category group that already exists in an organization, any category that is not defined in the XML file is permanently removed from your organization. For more information see Usage.

ObjectUsage

Represents the objects that can be associated with the data category group. This association allows the object to be classified and filtered using the data categories.

Field Name	Field Type	Description
`object`	string[]	A list of the object names that can be associated with the data category group. Valid values are: • `KnowledgeArticleVersion`—to associate articles. See "Modify Default Category Group Assignments for Articles" in the Salesforce online help for more information on data category groups association to articles. • `Question`—to associate questions. You can associate the `Question` object with at most one category group. See "Assigning Data Categories to Answers" in the Salesforce online help for more information on data category groups association to questions. Warning: If you deploy a category group that already exists in an organization, any object association that is not defined in the XML file is permanently removed from your organization. Ensure that your XML file specifies all the records associated with your category group in the organization. For more information see Usage.

Declarative Metadata Sample Definition

This sample is the definition of the `Geography` data category group and its data categories:

```
<?xml version="1.0" encoding="UTF-8"?>
<DataCategoryGroup xmlns="http://soap.sforce.com/2006/04/metadata">
   <label>Geography</label>
   <description>Geography structure of service center locations</description>
   <fullName>geo</fullName>

   <dataCategory> <name>WW</name> <label>Worldwide</label>
      <dataCategory> <name>AMER</name> <label>North America</label>
         <dataCategory>
             <name>USA</name>
             <label>United States of America</label>
         </dataCategory>
         <dataCategory>
             <name>CAN</name>
             <label>Canada</label>
         </dataCategory>
         <dataCategory>
             <name>MEX</name>
             <label>Mexico</label>
         </dataCategory>
      </dataCategory>
      <dataCategory> <name>EMEA</name> <label>Europe, Middle East, Africa</label>
         <dataCategory>
             <name>FR</name>
             <label>France</label>
         </dataCategory>
         <dataCategory>
             <name>SP</name>
             <label>Spain</label>
        </dataCategory>
         <dataCategory>
            <name>UK</name>
            <label>United-Kingdom</label>
         </dataCategory>
      </dataCategory>
      <dataCategory>
          <name>APAC</name>
          <label>Asia</label>
      </dataCategory>
   </dataCategory>

   <objectUsage>
      <object>KnowledgeArticleVersion </object>
   <objectUsage>
</DataCategoryGroup>
```

Usage

When you deploy a category group XML file, Metadata API checks whether the category group exists in the target organization. If the category group does not exist, it is created. If the category group already exists, then Metadata API:

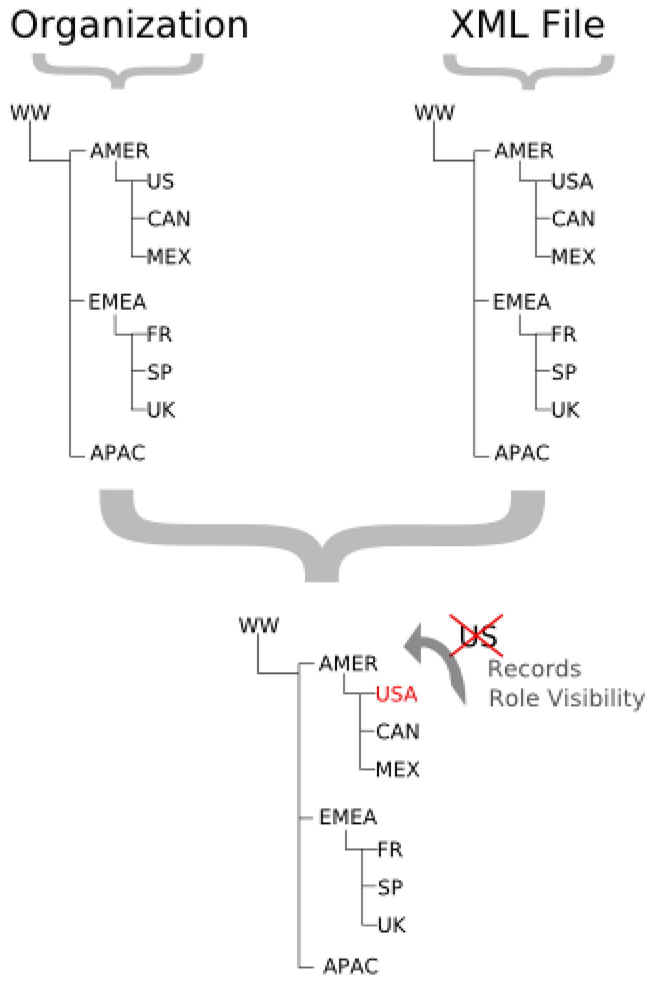
- Adds any new category or object defined in the XML file.
- Deletes any category that is not defined in the XML file. Records associated with the deleted categories are re-associated with the parent category.
- Deletes any object association that is not defined in the XML file.
- Moves any category if its hierarchical position differs from the position specified in the XML file.

  Note: When a category moves to a new parent category, users that have no visibility on the new parent category lose their visibility to the repositioned category.

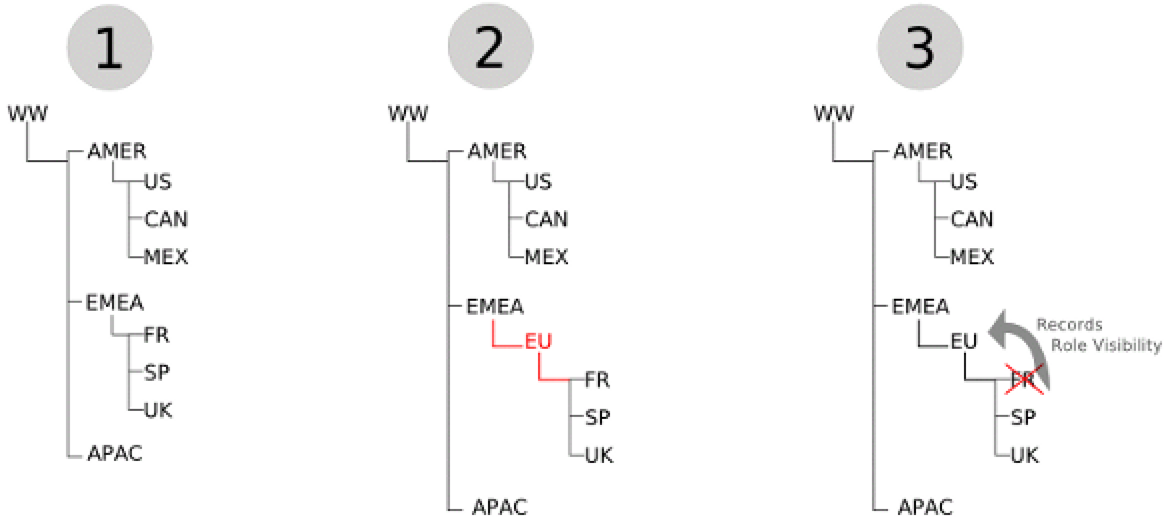
Note: For more information about category deletion, category repositioning and its impact on record categorization and visibility see "Delete a Data Category" and "Modify and Arrange Data Categories" in the Salesforce online help.

Using Metadata API to deploy category changes from one organization to another permanently removes categories and record categorizations that are not specified in your XML file. Salesforce recommends that you manually create data categories and record associations in an organization from Setup by entering *Data Categories* in the Quick Find box, then selecting **Data Categories** rather than deploying changes from a sandbox to a production organization.

The following example illustrates what happens if you deploy an XML representation of a Geography data category group hierarchy to an organization that already has this data category group defined. Note that the organization contains a US category, while the XML file includes a USA category in the same hierarchical position. The Metadata API deployment process deletes the US category from the organization and moves associations for any records from US to the parent AMER category. It also adds the USA category under AMER. Note that all records that were previously categorized with US are now associated with the AMER category.



The next example illustrates what can happen when you delete or move a category in a data category group and deploy its XML representation from a sandbox to a production organization that already has this data category group defined. Hierarchy 1 shows the initial data category group in the sandbox organization. In hierarchy 2, we add an `EU` category under `EMEA` and move `FR`, `SP` and `UK` below `EU`. In hierarchy 3, we delete `FR` and associate its records with its new parent, `EU`. Finally, we deploy the changes from the sandbox to the production organization.



Metadata API has no concept of the order of the changes made to the sandbox organization. It just deploys the changes from one organization to another. During the deployment, it first notices the deletion of the `FR` category and removes it from the production organization. Consequently, it moves associations for any records from `FR` to its parent on the production organization, `EMEA`. Metadata API then adds the `EU` category and moves `SP` and `UK` below it. Although the category group hierarchy looks the same in both organizations, record categorization in production is different from the sandbox organization. The records that were originally associated with `FR` in hierarchy 1 are associated with `EU` in the sandbox organization, but are associated with `EMEA` in the production organization.

Wildcard Support in the Manifest File

This metadata type supports the wildcard character `*` (asterisk) in the `package.xml` manifest file. For information about using the manifest file, see Deploying and Retrieving Metadata with the Zip File.

DelegateGroup

Represents a group of users who have the same administrative privileges. These groups are different from public groups used for sharing.

This type extends the Metadata metadata type and inherits its `fullName` field.

File Suffix and Directory Location

DelegateGroup components have the suffix `.delegateGroup` and are stored in the `delegateGroups` folder. The file prefix must match the developer name of the delegate group. For example, a delegate group with a developer name of MyDelegateGroup would have a file name of `MyDelegateGroup.delegateGroup`.

Version

DelegateGroup components are available in API version 36.0 and later.

Special Access Rules

Only users with the “View Setup and Configuration” permission can be delegated administrators. As of Spring ’20 and later, only users with “View Setup” or “Configuration” permission can access this object.

Fields

Field Name	Field Type	Description
`customObjects`	string[]	The custom objects associated with the group. Delegated administrators can customize nearly every aspect of each of those custom objects, including creating a custom tab. However, they cannot create or modify relationships on the objects or set organization-wide sharing defaults. Delegated administrators must have access to custom objects to access the merge fields on those objects from formulas.
`groups`	string[]	The groups with users assigned by delegated administrators.
`label`	string	Required. The delegated group’s non-API name.
`loginAccess`	boolean	Required. Allows users in this group to log in as users in the role hierarchy that they administer (`true`) or not (`false`). Depending on your organization settings, individual users must grant login access to allow their administrators to log in as them.
`permissionSets`	string[]	The permission sets assignable to users in specified roles and all subordinate roles by delegated administrators.
`profiles`	string[]	The profiles assignable to users by delegated administrators.
`roles`	string[]	The roles and subordinates for which delegated administrators of the group can create and edit users.

Declarative Metadata Sample Definition

The following is an example of a DelegateGroup component.

```xml
<?xml version="1.0" encoding="UTF-8"?>
<DelegateGroup xmlns="http://soap.sforce.com/2006/04/metadata">
    <label>MyDelegateGroup</label>
    <loginAccess>true</loginAccess>
    <name>MyDelegateGroup</name>
    <profiles>Chatter Free User</profiles>
    <profiles>Chatter Moderator User</profiles>
    <profiles>Marketing User</profiles>
    <permissionSets>My Permset</permissionSets>
    <roles>LesserBossMan</roles>
</DelegateGroup>
```

The following is an example `package.xml` that references the previous definition.

```xml
<?xml version="1.0" encoding="UTF-8"?>
<Package xmlns="http://soap.sforce.com/2006/04/metadata">
```

```
    <types>
        <members>*</members>
        <name>DelegateGroup</name>
    </types>
    <version>48.0</version>
</Package>
```

Wildcard Support in the Manifest File

This metadata type supports the wildcard character `*` (asterisk) in the `package.xml` manifest file. For information about using the manifest file, see Deploying and Retrieving Metadata with the Zip File.

Document

Represents a Document. All documents must be in a document folder, such as `sampleFolder/TestDocument`.

This type extends the MetadataWithContent metadata type and inherits its `content` and `fullName` fields.

Retrieving Documents

You can't use the wildcard (*) symbol with documents in `package.xml`. To retrieve the list of documents for populating `package.xml` with explicit names, call `listMetadata()` and pass in `DocumentFolder` as the type. Note that DocumentFolder is not returned as a type in `describeMetadata()`. Document is returned from `describeMetadata()` with an associated attribute of `inFolder` set to true. If that attribute is set to true, you can construct the type by using the component name with the word Folder, such as DocumentFolder.

The following example shows folders in `package.xml`:

```
<?xml version="1.0" encoding="UTF-8"?>
<Package xmlns="http://soap.sforce.com/2006/04/metadata">
    <types>
        <members>MyDBFolder/MyDBName</members>
        <name>Dashboard</name>
    </types>
    <types>
        <members>MyDocumentFolder/MyDocumentName</members>
        <name>Document</name>
    </types>
    <types>
        <members>unfiled$public/MarketingProductInquiryResponse</members>
        <members>unfiled$public/SalesNewCustomerEmail</members>
        <name>EmailTemplate</name>
    </types>
    <types>
        <members>MyReportFolder/MyReportName</members>
        <name>Report</name>
    </types>
    <version>48.0</version>
</Package>
```

For each document an accompanying metadata file named *DocumentFilename*-meta.xml is created in the document folder. For example, for a document TestDocument.png in the sampleFolder folder, there's a TestDocument.png-meta.xml in the documents/sampleFolder of the package.

Version

Documents are available in API version 10.0 and later.

In API version 17.0 and later, you can delete a folder containing documents moved to the Recycle Bin. When you delete the folder, any related documents in the Recycle Bin are permanently deleted.

In API version 18.0 and later, documents do not need an extension.

Fields

This metadata type contains the following fields:

Field Name	Field Type	Description
content	base64	Content of the document. Base 64-encoded binary data. Prior to making an API call, client applications must encode the binary attachment data as base64. Upon receiving a response, client applications must decode the base64 data to binary. This conversion is usually handled for you by a SOAP client. This field is inherited from the MetadataWithContent component.
description	string	A description of the document. Enter a description to distinguish this document from others.
fullName	string	The name of the document, including the folder name. In version 17.0 and earlier, the fullName included the document extension. In version 18.0 and later, the fullName does not include the file extension. The fullName can contain only underscores and alphanumeric characters. It must be unique, begin with a letter, not include spaces, not end with an underscore, and not contain two consecutive underscores. If this field contained characters before version 14.0 that are no longer allowed, the characters were stripped out of this field, and the previous value of the field was saved in the name field. This field is inherited from the Metadata component.
internalUseOnly	boolean	Required. Indicates whether the document is confidential (true) or not (false). This field and public are mutually exclusive; you cannot set both to true.
keywords	string	Contains one or more words that describe the document. A check for matches to words in this field is performed when doing a search.
name	string	The list of characters allowed in the fullName field has been reduced for versions 14.0 and later. This field contains the value contained in the fullName field before version 14.0. This field is only populated if the value of the fullName field contained characters that are no longer accepted in that field.

Field Name	Field Type	Description
`public`	boolean	Required. Indicates whether the document is an image available for HTML email templates and does not require a Salesforce username and password to view in an email (`true`) or not (`false`). If the images will be used as a custom app logo or custom tab icon, both of which require a Salesforce username and password to view, set this field to `false`. This field and `internalUseOnly` are mutually exclusive; you cannot set both to `true`.

Declarative Metadata Sample Definition

The following is the definition of a document:

```
<?xml version="1.0" encoding="UTF-8"?>
<Document xmlns="http://soap.sforce.com/2006/04/metadata">
    <internalUseOnly>false</internalUseOnly>
    <name>Q2 Campaign Analysis</name>
    <public>false</public>
    <description>Analyze Q2 campaign effectiveness</description>
</Document>
```

Wildcard Support in the Manifest File

This metadata type doesn't support the wildcard character `*` (asterisk) in the `package.xml` manifest file. For information about using the manifest file, see Deploying and Retrieving Metadata with the Zip File.

SEE ALSO:

Folder

DuplicateRule

Represents a rule that specifies how duplicate records in an object are detected. This type extends the Metadata metadata type and inherits its `fullName` field.

File Suffix and Directory Location

DuplicateRule components have the `.duplicateRule` suffix and are stored in the `duplicateRules/` directory. The name of the component file is based on the name of the object associated with the rule. For example, the component file name `duplicateRules/Account.Standard_Account_Duplicate_Rule.duplicateRule` describes a duplicate rule component associated with the Account object.

Version

DuplicateRule components are available in API version 48.0 and later.

Fields

Field Name	Field Type	Description
`actionOnInsert`	DupeActionType (enumeration of type string)	Required. Determines what the duplicate rule does when users or the DuplicateRule API try to insert a record that is a duplicate. Valid values are: **`Allow`** For users, if `operationsOnInsert` is set to `alert`, the UI displays the value of `alertText` in a dialog. The dialog prompts users to continue or cancel. If the user chooses to continue, the insertion proceeds. If the user chooses to cancel, the record isn't inserted. The DuplicateRule API returns an error code and a message. To complete the insertion, the code must set the `allowSave` field in DuplicateRuleHeader to `true` and reissue the request. If `operationsOnInsert`isn't set to `alert`, the UI inserts the record without issuing an alert. The API inserts the record and doesn't return an error code. **`Block`** For users, the UI displays an error message and prevents them from inserting the new record. The DuplicateRule API returns an error and doesn't insert the record.
`actionOnUpdate`	DupeActionType (enumeration of type string)	Required. Determines what the duplicate rule does when users or the DuplicateRule API try to update a record, and the result is a duplicate. Valid values are: **`Allow`** For users, if `operationsOnUpdate` is set to `alert`, the UI displays the value of `alertText` in a dialog. The dialog prompts users to continue or cancel. If the user chooses to continue, the update proceeds. If the user chooses to cancel, the record isn't updated. The DuplicateRule API returns a message. To complete the update, the code must set the `allowSave` field in DuplicateRuleHeader to `true` and reissue the request. If `operationsOnUpdate`isn't set to `alert`, the UI updates the record without issuing an alert. The API updates the record and doesn't return an error code. **`Block`** For users, the UI displays an error message and prevents them from continuing. The DuplicateRule API returns an error.

Field Name	Field Type	Description
`alertText`	string	Text that's sent when the duplicate rule is triggered. The text is only sent if `isActive` is `true`. In the UI, the text displays as a message. The DuplicateRule API returns the message in its response. You can set a value for `alertText` only when you have `actionOnInsert` or `actionOnUpdate` (or both) set to `Allow`. Otherwise, you receive a validation error when you add or update this component.
`description`	string	Required. Text that describes the duplicate rule. The value is customer-supplied, but is not visible in the UI.
`duplicateRuleFilter`	DuplicateRuleFilter	Required. Criteria that define how to find records to consider when looking for duplicates. For example, use `duplicateRuleFilter` to exclude records from the match when looking for duplicates.
`duplicateRuleMatchRules`	DuplicateRuleMatchRule[]	Required. One or more MatchingRule components for the DuplicateRule. A `MatchingRule` controls what constitutes a match between records.
`isActive`	boolean	Required. If `true`, the DuplicateRule detects duplicate records. Otherwise, the rule has no effect.
`masterLabel`	string	Required. Master label for this DuplicateRule. This value is the internal label for the rule.
`operationsOnInsert`	string[]	Required. Controls the action to take when `actionOnInsert` is set to `Allow` and the duplicate rule is triggered. Either one or both of these values can be set in the array: **`alert`** If set, the action specified in `actionOnInsert` occurs; otherwise, the insert proceeds. **`report`** If set, the insert operation is added to the report of duplicates.
`operationsOnUpdate`	string[]	Required. Controls the action to take when `actionOnUpdate` is set to `Allow` and the duplicate rule is triggered. Either one or both of these values can be set in the array: **`alert`** If set, the action specified in `actionOnUpdate` occurs; otherwise, the update proceeds. **`report`** If set, the update operation is added to the report of duplicates.
`securityOption`	DupeSecurityOptionType (enumeration of type string)	Required. Determines how record sharing rules affect duplicate management. Valid values are: **`EnforceSharingRules`** Sharing rules affect duplicate management. If a duplicate rule is triggered because an insert or update duplicates an existing record,

Field Name	Field Type	Description
		but the running user doesn't have sharing access to that record, the insert or update proceeds. The sharing rule doesn't prevent the user from creating or updating the record because the record is hidden from the user. No message is issued. **BypassSharingRules** Sharing rules don't affect duplicate management. If a duplicate rule is triggered because an insert or update duplicates an existing record, sharing rules are ignored, but other access restrictions apply.
sortOrder	int	Required. Determines the order in which duplicate rules are applied.

DuplicateRuleMatchRule

Describes the MatchingRule associated with the DuplicateRule. The MatchingRule identifies duplicate records.

Field Name	Field Type	Description
matchRuleSObjectType	string	Required. The name of the target object of the matching rule. For example, if you define a duplicate rule for Contact records, and you want to match with Lead records, the value of matchRuleSObjectType is Lead.
matchingRule	string	Required. Value that corresponds to the value of developerName in the MatchingRule for this duplicate rule.
objectMapping	ObjectMapping	Required. Foreign key to an ObjectMapping that maps fields from the duplicate rule's object to fields in the target object specified by matchRuleSObjectType.

DuplicateRuleFilter

Specifies filter criteria for a DuplicateRule. Salesforce only applies the DuplicateRule if the record matches the criteria.

Field Name	Field Type	Description
booleanFilter	string	Required. A string of boolean operators that establishes the filter logic for the filter items specified in duplicateRuleFilterItems.
duplicateRuleFilterItems	DuplicateRuleFilterItem[]	Required. A list of DuplicateRuleFilterItem components.

DuplicateRuleFilterItem

This type extends the FilterItem type and inherits all its fields.

Field Name	Field Type	Description
sortOrder	int	Required. The order of this item in the duplicate rule filter.

Field Name	Field Type	Description
`table`	string	Required. The object that has the field specified in the `field` field of DuplicateRuleFilterItem. See the documentation for FilterItem for the definition of `field`.

ObjectMapping

Represents a map of fields in the input object of the DuplicateRule to fields in the output object of DuplicateRule. The input object is the object associated with the DuplicateRule. The output object can be the same object or a different object with similar fields.

For example, you can have a DuplicateRule that looks for duplicates between the Contact object and the Lead object. In this case, the input object is Contact, and the output object is Lead.

Field Name	Field Type	Description
`inputObject`	string	Required. The input object for the duplicate rule. The DuplicateRule is associated with this object. For example, if you define a duplicate rule for Contact records, and you want to match with Lead records, the value of `inputObject` is Contact.
`mappingFields`	ObjectMappingField[]	Required. The mapping of source object fields to target object fields for the duplicate rule.
`outputObject`	string	Required. The output object for the duplicate rule. This value is the same as the value of the `matchRuleSObjectType` field in DuplicateRuleMatchRule. Any duplicate rules that this object has are ignored when the DuplicateRule API uses the ObjectMapping.

ObjectMappingField

A field name in the input object of the DuplicateRule, and the corresponding field name in the output object.

Field Name	Field Type	Description
`inputField`	string	Required. Field in the object specified by the `inputObject` field in ObjectMapping. This field is mapped to the field in `outputField`, which is assumed to be a field in the object specified by the `outputObject` field in ObjectMapping.
`outputField`	string	Required. Field in the object specified by the `outputObject` field in ObjectMapping. The field is mapped to the field name in `inputField`, which is assumed to be a field in the object specified by the `inputObject` in ObjectMapping.

Declarative Metadata Sample Definition

The following is an example of a DuplicateRule component.

```
<?xml version="1.0" encoding="UTF-8"?>
<DuplicateRule xmlns="http://soap.sforce.com/2006/04/metadata"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
    <actionOnInsert>Allow</actionOnInsert>
    <actionOnUpdate>Allow</actionOnUpdate>
    <alertText>You are creating a duplicate record. Use an existing record
instead.</alertText>
    <description>Detects a contact that duplicates a Lead</description>
    <duplicateRuleFilter>
        <booleanFilter xsi:nil="true"/>
        <duplicateRuleFilterItems>
            <field>Username</field>
            <operation>equals</operation>
            <value>user@example.com</value>
            <sortOrder>1</sortOrder>
            <table>User</table>
        </duplicateRuleFilterItems>
    </duplicateRuleFilter>
    <duplicateRuleMatchRules>
        <matchRuleSObjectType>Lead</matchRuleSObjectType>
        <matchingRule>ContactToLeadDuplicate_matching_rule</matchingRule>
        <objectMapping>
            <inputObject>Contact</inputObject>
            <mappingFields>
                <inputField>FirstName</inputField>
                <outputField>FirstName</outputField>
            </mappingFields>
            <mappingFields>
                <inputField>LastName</inputField>
                <outputField>LastName</outputField>
            </mappingFields>
            <outputObject>Lead</outputObject>
        </objectMapping>
    </duplicateRuleMatchRules>
    <isActive>true</isActive>
    <masterLabel>ContactToLeadDuplicate</masterLabel>
    <operationsOnInsert>Alert</operationsOnInsert>
    <operationsOnInsert>Report</operationsOnInsert>
    <operationsOnUpdate>Alert</operationsOnUpdate>
    <operationsOnUpdate>Report</operationsOnUpdate>
    <securityOption>EnforceSharingRules</securityOption>
    <sortOrder>1</sortOrder>
</DuplicateRule>
```

The following is an example `package.xml` that references the previous definition.

```
<?xml version="1.0" encoding="UTF-8"?>
<Package xmlns="http://soap.sforce.com/2006/04/metadata">
    <types>
        <members>ContactToLeadDuplicate</members>
        <name>DuplicateRule</name>
    </types>
    <version>38.0</version>
</Package>
```

Wildcard Support in the Manifest File

This metadata type supports the wildcard character `*` (asterisk) in the `package.xml` manifest file. For information about using the manifest file, see Deploying and Retrieving Metadata with the Zip File.

EclairGeoData

Represents an Analytics custom map chart. Custom maps are user-defined maps that are uploaded to Analytics and are used just as standard maps are. Custom maps are accessed in Analytics from the list of maps available with the map chart type.

File Suffix and Directory Location

EclairGeoData components have the suffix `geodata` and are stored in the `eclair` folder.

Version

EclairGeoData components are available in API version 39.0 and later.

Fields

Field Name	Field Type	Description
`maps`	EclairMap[]	A list of EclairMap objects. Each EclairMap object specifies the bounding box (if any) and the map name that appears in the user interface.
`masterLabel`	string	Required. Master label for this object. This display value is the internal label that is not translated.

EclairMap

Field Name	Field Type	Description
`boundingBoxBottom`	double	When bounding-box coordinates are used, this contains the bottom coordinate.
`boundingBoxLeft`	double	When bounding-box coordinates are used, this contains the left side coordinate.
`boundingBoxRight`	double	When bounding-box coordinates are used, this contains the right side coordinate.
`boundingBoxTop`	double	When bounding-box coordinates are used, this contains the top coordinate.
`mapLabel`	string	Required. The user-interface name of the map. This name appears in the maps list for the map chart in Analytics.

Field Name	Field Type	Description
mapName	string	Required. Label for this object. This display value is the internal label that is not translated.
projection	string	Required. The type of map projection used to create the map. Valid values are: • Equirectangular • Mercator • AlbersUSA

Declarative Metadata Sample Definition

The following is an example of an EclairGeoData component:

```
          <EclairGeoData xmlns="http://soap.sforce.com/2006/04/metadata"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
            <content xsi:nil="true"/>
            <maps>
              <boundingBoxBottom>0.0</boundingBoxBottom>
              <boundingBoxLeft>100.0</boundingBoxLeft>
              <boundingBoxRight>100.0</boundingBoxRight>
              <boundingBoxTop>0.0</boundingBoxTop>
              <mapLabel>WorldMap0 Label</mapLabel>
              <mapName>WorldMap0</mapName>
              <projection>Equirectangular</projection>
            </maps>
            <maps>
              <boundingBoxBottom>1.0</boundingBoxBottom>
              <boundingBoxLeft>101.0</boundingBoxLeft>
              <boundingBoxRight>101.0</boundingBoxRight>
              <boundingBoxTop>1.0</boundingBoxTop>
              <mapLabel>WorldMap1 Label</mapLabel>
              <mapName>WorldMap1</mapName>
              <projection>Mercator</projection>
            </maps>
            <masterLabel>WorldMapGeoDataToCreate Label</masterLabel>
          </EclairGeoData>
```

The following is an example package.xml that references the previous definition.

```
        <?xml version="1.0" encoding="UTF-8"?>
        <Package xmlns="http://soap.sforce.com/2006/04/metadata">
          <types>
            <members>*</members>
            <name>EclairGeoData</name>
          </types>
          <version>39.0</version>
        </Package>
```

Wildcard Support in the Manifest File

This metadata type supports the wildcard character `*` (asterisk) in the `package.xml` manifest file. For information about using the manifest file, see Deploying and Retrieving Metadata with the Zip File.

EmailServicesFunction

Represents an email service. This type extends the Metadata metadata type and inherits its `fullName` field.

File Suffix and Directory Location

EmailServicesFunction components have the suffix `.xml` and are stored in the `emailservices` folder.

Version

EmailServicesFunction components are available in API version 42.0 and later.

Fields

Field Name	Field Type	Description
`apexClass`	string	Required. The name of the Apex class that the email service uses to process inbound messages.
`attachmentOption`	EmailServicesAttOptions (enumeration of type string)	Required. Indicates the types of attachments the email service accepts. One of the following values: • `None`—The email service accepts the message but discards any attachment. • `NoContent`—The attachment metadata (filename, MIME type, and so on) is provided to the Apex class, but the body is set to `null`. • `TextOnly`—The email service only accepts the following types of attachments: – Attachments with a Multipurpose Internet Mail Extension (MIME) type of text. – Attachments with a MIME type of application/octet-stream and a file name that ends with either a .vcf or .vcs extension. These are saved as text/x-vcard and text/calendar MIME types, respectively. • `BinaryOnly`—The email service only accepts binary attachments, such as image, audio, application, and video files. • `All`—The email service accepts any type of attachment.

Field Name	Field Type	Description
`authenticationFailureAction`	EmailServicesErrorAction (enumeration of type string)	Required. Indicates what the email service does with messages that fail or do not support any of the authentication protocols if the `isAuthenticationRequired` field is true. One of the following values: • `UseSystemDefault`—The system default is used. • `Bounce`—The email service returns the message to the sender with a notification that explains why the message was rejected. • `Discard`—The email service deletes the message without notifying the sender. • `Requeue`—The email service queues the message for processing in the next 24 hours. If the message is not processed within 24 hours, the email service returns the message to the sender with a notification that explains why the message was rejected.
`authorizationFailureAction`	EmailServicesErrorAction (enumeration of type string)	Required. Indicates what the email service does with messages received from senders who are not listed in the `authorizedSenders` field on either the email service or email service address. One of the following values: • `UseSystemDefault`—The system default is used. • `Bounce`—The email service returns the message to the sender with a notification that explains why the message was rejected. • `Discard`—The email service deletes the message without notifying the sender. • `Requeue`—The email service queues the message for processing in the next 24 hours. If the message is not processed within 24 hours, the email service returns the message to the sender with a notification that explains why the message was rejected.
`authorizedSenders`	string	Configures the email service to only accept messages from the email addresses or domains listed in this field. If the email service receives a message from an unlisted email address or domain, the email service performs the action specified in the `authorizationFailureAction` field. Leave this field blank if you want the email service to receive email from any email address.
`emailServicesAddresses`	EmailServicesAddress[]	A list of EmailServiceAddress records.
`errorRoutingAddress`	email	The destination email address for error notification email messages when `isErrorRoutingEnabled` is `true`.
`functionInactiveAction`	EmailServicesErrorAction (enumeration of type string)	Required. Indicates what the email service does with messages it receives when the email service itself is inactive. One of the following values: • `UseSystemDefault`—The system default is used.

Field Name	Field Type	Description
		• `Bounce`—The email service returns the message to the sender with a notification that explains why the message was rejected. • `Discard`—The email service deletes the message without notifying the sender. • `Requeue`—The email service queues the message for processing in the next 24 hours. If the message is not processed within 24 hours, the email service returns the message to the sender with a notification that explains why the message was rejected.
`functionName`	string	Required. The name of the email service in the API. This name can contain only underscores and alphanumeric characters and must be unique in your org. The value in this 64-character field must begin with a letter, not include spaces, not end with an underscore, and not contain two consecutive underscores. In managed packages, this field prevents naming conflicts on package installations. This field is automatically generated, but you can supply your own value if you create the record using the API. With this field, a developer can change the object's name in a managed package and the changes are reflected in a subscriber's organization. Note: When creating large sets of data, always specify a unique `functionName` for each record. If no `functionName` is specified, performance may slow while Salesforce generates one for each record.
`isActive`	boolean	Indicates whether this object is active (`true`) or not (`false`).
`isAuthenticationRequired`	boolean	Configures the email service to verify the legitimacy of the sending server before processing a message. The email service uses the SPF, SenderId, and DomainKeys protocols to verify the sender's legitimacy: If the sending server passes at least one of these protocols and does not fail any, the email service accepts the email. If the server fails a protocol or does not support any of the protocols, the email service performs the action specified in the `authenticationFailureAction` field.
`isErrorRoutingEnabled`	boolean	When incoming email messages can't be processed, indicates whether error notification email messages are routed to a chosen address or to the senders.
`isTextAttachmentsAsBinary`	boolean	If `true`, text attachments are supplied to the Apex code as a `Messaging.BinaryAttachment` instead of as a `Messaging.TextAttachment`. This means that the body is supplied as an Apex Blob instead of as an Apex String.
`isTlsRequired`	boolean	Not currently in use.

Field Name	Field Type	Description
`overLimitAction`	EmailServicesErrorAction (enumeration of type string)	Required. Indicates what the email service does with messages if the total number of messages processed by all email services combined has reached the daily limit for your organization. One of the following values: • `UseSystemDefault`—The system default is used. • `Bounce`—The email service returns the message to the sender with a notification that explains why the message was rejected. • `Discard`—The email service deletes the message without notifying the sender. • `Requeue`—The email service queues the message for processing in the next 24 hours. If the message is not processed within 24 hours, the email service returns the message to the sender with a notification that explains why the message was rejected. The system calculates the limit by multiplying the number of user licenses by 1,000.

EmailServicesAddress

Each email service has one or more email addresses to which users can send messages for processing. An email service only processes messages it receives at one of its addresses.

Field Name	Field Type	Description
`authorizedSenders`	string	Configures the email service address to only accept messages from the email addresses or domains listed in this field. If the email service address receives a message from an unlisted email address or domain, the email service performs the action specified in the `authorizationFailureAction` field of its associated email service. Leave this field blank if you want the email service address to receive email from any email address.
`developerName`	string	Required. The name of the object in the API. This name can contain only underscores and alphanumeric characters and must be unique in your org. It must begin with a letter, not include spaces, not end with an underscore, and not contain two consecutive underscores. This 25-character field must be unique among other EmailServicesAddress records under the same EmailServiceFunction parent. In managed packages, this field prevents naming conflicts on package installations. This field is automatically generated, but you can supply your own value if you create the record using the API. With this field, a developer can change the object's name in a managed package and the changes are reflected in a subscriber's organization. Note: When creating large sets of data, always specify a unique `developerName` for each record. If no `developerName` is

Field Name	Field Type	Description
		specified, performance might be slow while Salesforce generates one for each record.
`isActive`	boolean	Indicates whether this object is active (`true`) or not (`false`).
`localPart`	string	Required. The local-part of the email service address, which is the string that comes before the @ symbol. For the local-part of a Salesforce email address, all alphanumeric characters are valid, plus the following special characters: `! # $ % & amp; ' * / = ? ^ _ + - ` { | } ~ ,` The dot character (.) is also valid as long as it's not the first or last character. Email addresses aren't case sensitive.
`runAsUser`	string	Required. The username of the user whose permissions the email service assumes when processing messages sent to this address.

Wildcard Support in the Manifest File

This metadata type doesn't support the wildcard character `*` (asterisk) in the `package.xml` manifest file. For information about using the manifest file, see Deploying and Retrieving Metadata with the Zip File.

EmailTemplate

Represents an email template.

This type extends the MetadataWithContent metadata type and inherits its `content` and `fullName` fields.

Note: Packaging isn't supported for Lightning email templates.

File Suffix and Directory Location

The file suffix is `.email` for the template file. The accompanying metadata file is named *`EmailTemplateName`*`-meta.xml`.

EmailTemplate components are stored in the `email` folder in the corresponding package directory. For example, for an email template named SampleTemplate in the sampleFolder folder, there's a `SampleTemplate-meta.xml` in the `email/sampleFolder` of the package.

Retrieving Email Templates

You can't use the wildcard (*) symbol with email templates in `package.xml`. To retrieve the list of email templates for populating `package.xml` with explicit names, call `listMetadata()` and pass in `EmailTemplate` as the type.

The following example shows folders in `package.xml`:

```
<?xml version="1.0" encoding="UTF-8"?>
<Package xmlns="http://soap.sforce.com/2006/04/metadata">
    <types>
        <members>MyDBFolder/MyDBName</members>
        <name>Dashboard</name>
    </types>
    <types>
        <members>MyDocumentFolder/MyDocumentName</members>
        <name>Document</name>
    </types>
    <types>
        <members>unfiled$public/MarketingProductInquiryResponse</members>
        <members>unfiled$public/SalesNewCustomerEmail</members>
        <name>EmailTemplate</name>
    </types>
    <types>
        <members>MyReportFolder/MyReportName</members>
        <name>Report</name>
    </types>
    <version>48.0</version>
</Package>
```

Version

Email templates are available in API version 12.0 and later.

Fields

This metadata type contains the following fields:

Field Name	Field Type	Description
apiVersion	double	The API version if this is a Visualforce email template. Every Visualforce email template has an API version specified at creation. This field is available in API version 16.0 and later.
attachedDocuments	string[]	A list of references to documents in your organization. These documents are included as attachments in the email template. Each document is referenced by its path, for example MyFolder/MyDocument.txt.
attachments	Attachment[]	A list of attachments for the email template.
available	boolean	Required. Indicates whether this template is offered to users when sending an email (true) or not (false).
content	base64Binary	Content of the email template. Base 64-encoded binary data. Prior to making an API call, client applications must encode the binary attachment data as base64. Upon receiving a response, client applications must decode the base64 data to binary. This conversion is usually handled for you by a SOAP client. This field contains: • Binary content of the email body if type is set to text • HTML email content if type is set to html • HTML body if type is set to custom • Visualforce body if type is set to visualforce

Field Name	Field Type	Description
		This field is inherited from the MetadataWithContent component.
`description`	string	The email template description. This can be useful to describe the reason for creating the template.
`encodingKey`	Encoding (enumeration of type string)	Required for Classic email templates. The default encoding setting is Unicode: `UTF-8`. Change it if your template requires data in a different format. Valid values include: • `UTF-8`—Unicode (UTF-8) • `ISO-8859-1`—General US & Western Europe (ISO-8859–1, ISO-LATIN-1) • `Shift_JIS`—Japanese (Shift-JIS) • `ISO-2022-JP`—Japanese (JIS) • `EUC-JP`—Japanese (EUC-JP) • `x-SJIS_0213`—Japanese (Shift-JIS_2004) • `ks_c_5601-1987`—Korean (ks_c_5601-1987) • `Big5`—Traditional Chinese (Big5) • `GB2312`—Simplified Chinese (GB2312) • `Big5-HKSCS`—Traditional Chinese Hong Kong (Big5–HKSCS) Note: Lightning email templates don't use this field. Instead, the encoding values are taken directly from the user's encoding settings.
`fullName`	string	The email template developer name used as a unique identifier for API access. The `fullName` can contain only underscores and alphanumeric characters. It must be unique, begin with a letter, not include spaces, not end with an underscore, and not contain two consecutive underscores. If this field contained characters before version 14.0 that are no longer allowed, the characters were stripped out of this field, and the previous value of the field was saved in the `name` field. This field is inherited from the Metadata component.
`letterhead`	string	The letterhead name associated with this email template. Only available when `type` is set to `html`.
`name`	string	Required. Email template name. The list of characters allowed in the `fullName` field has been reduced for versions 14.0 and later. This field contains the value contained in the `fullName` field before version 14.0.
`packageVersions`	PackageVersion[]	The list of package versions for any managed packages containing components that are referenced by this email template. This field is only relevant for Visualforce email templates. For more information about managed packages, see the Lightning Platform Quick Reference for Developing Packages. For more information about package versions, see "About Package Versions" in the Salesforce online help. This field is available in API version 16.0 and later.

Field Name	Field Type	Description
`relatedEntityType`	Object Name (enumeration of type string)	Reserved for future use with Lightning Experience.
`style`	EmailTemplateStyle (enumeration of type string)	Required. The style of the template. This field is only available when `type` is set to `html`. Valid style values include: • `none` • `freeForm` • `formalLetter` • `promotionRight` • `promotionLeft` • `newsletter` • `products`
`subject`	string	The email subject.
`textOnly`	string	The text of the email body if `type` is set to `html` or `custom`.
`type`	EmailTemplateType (enumeration of type string)	Required. The email template type. The valid values are: • `text` -all users can create or change text email templates. • `html` - administrators and users with the “Edit HTML Templates” permission can create HTML email templates based on a letterhead. • `custom` - administrators and users with the “Edit HTML Templates” permission can create custom HTML email templates without using a letterhead. You must either know HTML or obtain the HTML code to insert in your email template. • `visualforce` - administrators and users with the “Customize Application” permission can create email templates using Visualforce.
`UiType`	EmailTemplateUiType (enumeration of type string)	Indicates the user interface where this template is usable. Valid values are: • `Aloha` (Salesforce Classic) • `SFX` (Lightning Experience) • `SFX_Sample` (Lightning Experience Sample) If `UiType` is `SFX`, the `type` must be `custom`. Note: Packaging is supported for Salesforce Classic email templates only.

Example:

```
<EmailTemplate>
```

```
    <available>true</available>
    <description>Notification that user has been added to a community.</description>
    <encodingKey>UTF-8</encodingKey>
    <name>Communities: New Member Welcome Email</name>
    <style>none</style>
    <subject>Welcome to {!Community_Name}</subject>
    <type>custom</type>
    <uiType>Aloha</uiType>
</EmailTemplate>
```

Attachment

Attachment represents an email attachment.

Field	Field Type	Description
`content`	base64Binary	Required. The attachment content. Base 64-encoded binary data. Prior to making an API call, client applications must encode the binary attachment data as base64. Upon receiving a response, client applications must decode the base64 data to binary. This conversion is usually handled for you by a SOAP client.
`name`	string	Required. The attachment file name.

Declarative Metadata Sample Definition

A sample XML definition of an email template is shown below.

```
<?xml version="1.0" encoding="UTF-8"?>
<EmailTemplate xmlns="http://soap.sforce.com/2006/04/metadata">
    <available>true</available>
    <description>Sample Email Template</description>
    <encodingKey>ISO-8859-1</encodingKey>
    <name>Sample Email Template</name>
    <style>none</style>
    <subject>Sample email subject</subject>
    <textOnly>Your case has been resolved.</textOnly>
    <type>custom</type>
</EmailTemplate>
```

Wildcard Support in the Manifest File

This metadata type doesn't support the wildcard character `*` (asterisk) in the `package.xml` manifest file. For information about using the manifest file, see Deploying and Retrieving Metadata with the Zip File.

SEE ALSO:

Letterhead

EmbeddedServiceBranding

Represents the branding for each Embedded Service deployment. This type extends the Metadata metadata type and inherits its `fullName` field.

File Suffix and Directory Location

EmbeddedServiceBranding components are stored in the *developer_name*`.EmbeddedServiceBranding` file in the `EmbeddedServiceBranding` folder.

Version

EmbeddedServiceBranding is available in API version 39.0 and later.

Fields

Field Name	Field Type	Description
`contrastInvertedColor`	string	Accent branding color used in the embedded component, displayed as a hexadecimal value. Changes made to this field in the API aren't reflected in the embedded component.
`contrastPrimaryColor`	string	Accent branding color used in the embedded component, displayed as a hexadecimal value.
`embeddedServiceConfig`	string	Required. The Embedded Service configuration that this branding applies to.
`font`	string	Font used in the text of the embedded component.
`height`	int	Height of the embedded component. Available in API version 43.0 and later.
`masterLabel`	string	Required. The name of the Embedded Service configuration node.
`navBarColor`	string	Color used for the navigation bar in the embedded component, displayed as a hexadecimal value.
`primaryColor`	string	Primary branding color used in the embedded component, displayed as a hexadecimal value.
`secondaryColor`	string	Secondary branding color used in the embedded component, displayed as a hexadecimal value.
`width`	int	Width of the embedded component. Available in API version 43.0 and later.

Declarative Metadata Sample Definition

The following is an example of an EmbeddedServiceBranding file.

```
<?xml version="1.0" encoding="UTF-8"?>
<EmbeddedServiceBranding xmlns="http://soap.sforce.com/2006/04/metadata">
    <contrastInvertedColor>#ffffff</contrastInvertedColor>
    <contrastPrimaryColor>#333333</contrastPrimaryColor>
    <embeddedServiceConfig>EswConfig001</embeddedServiceConfig>
    <font>Salesforce Sans</font>
    <height>498</height>
    <masterLabel>EmbeddedServiceBranding_Parent04IRM000000002a_16033cd2c16</masterLabel>
    <navBarColor>#222222</navBarColor>
    <primaryColor>#222222</primaryColor>
    <secondaryColor>#005290</secondaryColor>
    <width>320</width>
</EmbeddedServiceBranding>
```

Wildcard Support in the Manifest File

This metadata type doesn't support the wildcard character * (asterisk) in the `package.xml` manifest file. For information about using the manifest file, see Deploying and Retrieving Metadata with the Zip File.

EmbeddedServiceConfig

Represents a setup node for creating an Embedded Service for Web deployment. This type extends the Metadata metadata type and inherits its `fullName` field.

File Suffix and Directory Location

EmbeddedServiceConfig components are stored in the *`developer_name`*`.EmbeddedServiceConfig` file in the `EmbeddedServiceConfig` folder.

Version

EmbeddedServiceConfig is available in API version 37.0 and later.

Fields

Field Name	Field Type	Description
`areGuestUsersAllowed`	boolean	Specifies whether a user must be logged in to access an embedded component. Available in API version 45.0 and later.
`authMethod`	EmbeddedServiceAuthMethod (enumeration of type string)	Type of login method selected for this Embedded Service deployment. One of the following values: • `CommunitiesLogin`—Customers log in using Communities.

Field Name	Field Type	Description
		• CustomLogin—Customers log in using your own custom authentication. Available in API version 43.0 and later.
customMinimizedComponent	string	The custom Lightning component that's used in this Embedded Service deployment in its minimized state. Available in API version 43.0 and later.
embeddedServiceCustomComponents	EmbeddedServiceCustomComponent on page 463[]	The custom components used in this Embedded Service deployment. Available in API version 44.0 and later.
embeddedServiceCustomLabels	EmbeddedServiceCustomLabel on page 464[]	The custom labels used in this Embedded Service deployment. Available in API version 44.0 and later.
embeddedServiceFlowConfig	EmbeddedServiceFlowConfig on page 464[]	Represents a setup node for creating an embedded flow. Available in API version 45.0 and later.
embeddedServiceFlows	EmbeddedServiceFlow on page 464[]	All of the flows used by this Embedded Service deployment. Available in API version 45.0 and later.
masterLabel	string	Required. The name of the Embedded Service configuration node. Available in API version 37.0 and later.
shouldHideAuthDialog	boolean	Specifies whether the prompt that the customer login again during a flow is hidden (true) or not (false). When it's hidden, the customer is taken directly to your login page. This field is set to false by default. Available in API version 43.0 and later.
site	string	Required. The name of the Salesforce Community or site connected to this Embedded Service deployment. Available in API version 37.0 and later.

EmbeddedServiceCustomComponent

Returns a custom component that's associated with an EmbeddedServiceConfig setup.

Field Name	Field Type	Description
customComponent	string	The name of the custom component.
customComponentType	EmbeddedServiceCustomComponent (enumeration of type string)	The type of custom component. One of the following values: • LA_Prechat (component for pre-chat in Embedded Chat) • LA_Minimized (component for the minimized chat window) • LA_PlainTextChatMessage (component for the text area in Embedded Chat)

EmbeddedServiceCustomLabel

Returns a custom label that's associated with an EmbeddedServiceConfig setup.

Field Name	Field Type	Description
`customLabel`	string	The customized label that appears in the embedded component.
`feature`	EmbeddedServiceFeature (enumeration of type string)	The feature that this embedded component uses. One of the following values: • `Base` • `FieldService` • `Flows` • `LiveAgent` • `NotInUse`
`labelKey`	EmbeddedServiceLabelKey (enumeration of type string)	The type of label for this embedded component. The value corresponds to the label within a label group (substate of chat state or page type).

EmbeddedServiceFlow

Returns an embedded flow that's associated with an EmbeddedServiceConfig setup.

Field Name	Field Type	Description
`flow`	string	The developer name of the flow.
`flowType`	EmbeddedServiceFlowType(enumeration of type string)	The type of flow. One of the following values: • `FS_Flow` • `FS_NewAppointment` • `FS_ModifyAppointment` • `FS_CancelAppointment`
`isAuthenticationRequired`	boolean	Indicates whether users are required to log in to access the Embedded Service component. The value cannot be `true` for the `FS_Flow` value and must be `true` for all other values.

EmbeddedServiceFlowConfig

Returns the EmbeddedServiceFlowConfig type.

Field Name	Field Type	Description
`enabled`	boolean	Indicates whether the embedded flow is enabled.

Declarative Metadata Sample Definition

The following is an example of an EmbeddedServiceConfig file.

```
This is an example of an EmbeddedServiceConfig file.
<?xml version="1.0" encoding="UTF-8"?>
<EmbeddedServiceConfig xmlns="http://soap.sforce.com/2006/04/metadata">
    <authMethod>CommunitiesLogin</authMethod>
    <customMinimizedComponent>customMinimized</customMinimizedComponent>
    <masterLabel>EswFS</masterLabel>
    <shouldHideAuthDialog>false</shouldHideAuthDialog>
    <site>SiteName</site>
</EmbeddedServiceConfig>
```

Wildcard Support in the Manifest File

This metadata type doesn't support the wildcard character `*` (asterisk) in the `package.xml` manifest file. For information about using the manifest file, see Deploying and Retrieving Metadata with the Zip File.

EmbeddedServiceFieldService

Represents a setup node for creating an embedded Appointment Management deployment. This type extends the Metadata metadata type and inherits its `fullName` field.

File Suffix and Directory Location

EmbeddedServiceFieldService components are stored in a *`developer_name`*`.EmbeddedServiceFieldService` file in the `EmbeddedServiceFieldService` folder.

Version

EmbeddedServiceFieldService is available in API version 43.0 and later.

Fields

Field Name	Field Type	Description
`appointmentBookingFlowName`	string	Name of the appointment booking flow for this embedded Appointment Management (beta) deployment.
`cancelApptBookingFlowName`	string	Name of the appointment cancellation flow for this embedded Appointment Management (beta) deployment.

Field Name	Field Type	Description
embeddedServiceConfig	string	Required. The name of the Embedded Service configuration node.
enabled	boolean	Required. Indicates whether this embedded Appointment Management deployment is enabled (true).
fieldServiceConfirmCardImg	string	URL of the image used for the confirmation card in embedded Appointment Management (beta).
fieldServiceHomeImg	string	URL of the image used for the home screen in embedded Appointment Management (beta).
fieldServiceLogoImg	string	URL of the logo used for the home screen in embedded Appointment Management (beta).
masterLabel	string	Required. Name of the embedded Appointment Management (beta) deployment.
modifyApptBookingFlowName	string	Name of the appointment modification flow for this embedded Appointment Management (beta) deployment.
shouldShowExistingAppointment	boolean	Specifies whether to display a button on the home screen for customers to access their existing appointments (true) or not (false). This field is false by default.
shouldShowNewAppointment	boolean	Specifies whether to display a button on the home screen for customers to create a new appointment (true) or not (false). This field is false by default.

Declarative Metadata Sample Definition

The following is an example of an EmbeddedServiceFieldService file.

```
<?xml version="1.0" encoding="UTF-8"?>
<EmbeddedServiceFieldService xmlns="http://soap.sforce.com/2006/04/metadata">
    <appointmentBookingFlowName>ESW_FS_BookAppt_Main_Flow</appointmentBookingFlowName>
    <cancelApptBookingFlowName>ESW_FS_CancelAppt_Flow</cancelApptBookingFlowName>
    <embeddedServiceConfig>EswFS</embeddedServiceConfig>
    <enabled>true</enabled>

<fieldServiceConfirmCardImg>https://google.com/AppointmentConfirmationImg.png</fieldServiceConfirmCardImg>

    <fieldServiceHomeImg>https://google.com/HeroImg.png</fieldServiceHomeImg>
    <fieldServiceLogoImg>https://google.com/logo.png</fieldServiceLogoImg>

<masterLabel>EmbeddedServiceFieldService_Parent04IRM000000007p2AA_162d4270834</masterLabel>

    <modifyApptBookingFlowName>ESW_FS_ModifyAppt_Main_Flow</modifyApptBookingFlowName>
    <shouldShowExistingAppointment>true</shouldShowExistingAppointment>
    <shouldShowNewAppointment>true</shouldShowNewAppointment>
</EmbeddedServiceFieldService>
```

Usage

Note: Any changes you make to the image fields override what you've entered in Setup. We recommend setting your image URLs in Setup.

EmbeddedServiceFlowConfig

Represents a setup node for creating an embedded flow. This type extends the Metadata metadata type and inherits its `fullName` field.

File Suffix and Directory Location

EmbeddedServiceFlowConfig components are stored in the *`developer_name`*`.EmbeddedServiceFlowConfig` file in the `EmbeddedServiceFlowConfig` folder.

Version

EmbeddedServiceFlowConfig is available in API version 45.0 and later.

Fields

Field Name	Field Type	Description
`enabled`	boolean	Indicates whether the embedded flow is enabled (`true`) or not (`false`). Defaults to `false`.

Declarative Metadata Sample Definition

The following is an example of an EmbeddedServiceFlowConfig file.

```
<?xml version="1.0" encoding="UTF-8"?>
<EmbeddedServiceFlowConfig xmlns="http://soap.sforce.com/2006/04/metadata">
    <enabled>true</enabled>
</EmbeddedServiceFlowConfig>
```

EmbeddedServiceLiveAgent

Represents a setup node for creating an embedded chat deployment. This type extends the Metadata metadata type and inherits its `fullName` field.

File Suffix and Directory Location

EmbeddedServiceLiveAgent components are stored in the *`developer_name`*`.EmbeddedServiceLiveAgent` file in the `EmbeddedServiceLiveAgent` folder.

Version

EmbeddedServiceLiveAgent is available in API version 38.0 and later.

Fields

Field Name	Field Type	Description
avatarImg	string	Avatar image for this embedded chat deployment.
customPrechatComponent	string	The custom Lightning Component that's used for the pre-chat page in this embedded chat deployment.
embeddedServiceConfig	string	Required. The name of the embedded service configuration node.
embeddedServiceQuickActions	EmbeddedServiceQuickAction	The quick action used by the pre-chat form.
enabled	boolean	Required. Indicates whether this embedded chat deployment is enabled (true).
fontSize	EmbeddedServiceFontSize (enumeration of type string)	Required. The font size for the text in the embedded chat window. One of the following values: - Small - Medium - Large
headerBackgroundImg	string	Header background image for this embedded chat window.
isOfflineCaseEnabled	boolean	Indicates whether offline support is enabled for this embedded chat deployment. Available in API version 43.0 and later.
isQueuePositionEnabled	boolean	Indicates whether queue position (displaying the chat visitor's place in line while they wait for an agent) is enabled for this embedded chat deployment. Available in API version 43.0 and later.
liveAgentChatUrl	string	The rest endpoint for chats.
liveAgentContentUrl	string	The rest endpoint for cChat content.
liveChatButton	string	Required. Reference to a chat button created in Chat setup.
liveChatDeployment	string	Required. Reference to a deployment created in Chat setup.
masterLabel	string	Required. Name of the embedded chat deployment.
offlineCaseBackgroundImg	string	Offline support case form background image for this embedded chat window. Available in API version 43.0 and later.
prechatBackgroundImg	string	Pre-chat background image for this embedded chat window.
prechatEnabled	string	Required. Indicates whether the embedded chat pre-chat form is enabled for this deployment.
prechatJson	string	JSON object of all the fields of the selected pre-chat form in Chat setup.

Field Name	Field Type	Description
`scenario`	EmbeddedServiceScenario (enumeration of type string)	Required. The scenario for the embedded chat window that determines which objects to relate to the chat. One of the following values: • `Sales` • `Service` • `Basic`
`smallCompanyLogoImg`	string	Company logo image for this embedded chat window.
`waitingStateBackgroundImg`	string	Chat waiting image for this embedded chat window.

EmbeddedServiceQuickAction

Returns a quick action that's associated with an EmbeddedServiceLiveAgent setup. The quick action include the pre-chat form fields that the embedded chat window displays and shows the order in which he fields are displayed.

Field Name	Field Type	Description
`embeddedServiceLiveAgent`	string	Reference to the embedded chat deployment.
`order`	int	Order in which this quick action appears in the embedded chat pre-chat form.
`quickActionDefinition`	string	Reference to a quick action.
`quickActionType`	EmbeddedServiceQuickActionType (enumeration of type string)	Quick action type. One of the following values: • `Prechat`—Pre-chat • `OfflineCase`—Offline support (Cases) Available in API version 43.0 and later.

Declarative Metadata Sample Definition

The following is an example of an EmbeddedServiceLiveAgent file.

```
<?xml version="1.0" encoding="UTF-8"?>
<EmbeddedServiceLiveAgent xmlns="http://soap.sforce.com/2006/04/metadata">
    <avatarImg>https://google.com/avatar.png</avatarImg>
    <customPrechatComponent>auraCustomPrechat</customPrechatComponent>
    <embeddedServiceConfig>EswConfig001</embeddedServiceConfig>
    <embeddedServiceQuickActions>

<embeddedServiceLiveAgent>EmbeddedServiceLiveAgent_Parent04Ixx0000000001EAA_15ec5bd2971</embeddedServiceLiveAgent>

        <order>1</order>

<quickActionDefinition>Snapins_Contact_QuickAction_08hRM000000001h</quickActionDefinition>

    </embeddedServiceQuickActions>
    <embeddedServiceQuickActions>
```

```
<embeddedServiceLiveAgent>EmbeddedServiceLiveAgent_Parent04Ixx0000000001EAA_15ec5bd2971</embeddedServiceLiveAgent>

        <order>1</order>

<quickActionDefinition>Snapins_Case_OfflineCaseQuickAction_08hRM000000001h</quickActionDefinition>

        <quickActionType>OfflineCase</quickActionType>
    </embeddedServiceQuickActions>
    <embeddedServiceQuickActions>

<embeddedServiceLiveAgent>EmbeddedServiceLiveAgent_Parent04Ixx0000000001EAA_15ec5bd2971</embeddedServiceLiveAgent>

        <order>2</order>

<quickActionDefinition>Snapins_Case_QuickAction_08hRM000000001h</quickActionDefinition>
    </embeddedServiceQuickActions>
    <enabled>true</enabled>
    <fontSize>Medium</fontSize>
    <headerBackgroundImg>https://google.com/headerBackgroundImg.png</headerBackgroundIm>
    <isOfflineCaseEnabled>true</isOfflineCaseEnabled>
    <isQueuePositionEnabled>true</isQueuePositionEnabled>
    <liveChatButton>chatButton01</liveChatButton>
    <liveChatDeployment>liveAgentDeployment01</liveChatDeployment>
   <masterLabel>EmbeddedServiceLiveAgent_Parent04Ixx0000000001EAA_15ec5bd2971</masterLabel>


<offlineCaseBackgroundImg>https://google.com/offlineCaseBackgroundImg.png</offlineCaseBackgroundImg>

   <prechatBackgroundImg>https://google.com/prechatBackgroundImg.png</prechatBackgroundImg>

    <prechatEnabled>true</prechatEnabled>
    <scenario>Service</scenario>
    <smallCompanyLogoImg>https://google.com/smallCompanyLogoImg.png</smallCompanyLogoImg>


<waitingStateBackgroundImg>https://google.com/waitingImage.png</waitingStateBackgroundImg>
</EmbeddedServiceLiveAgent>
```

Usage

EmbeddedServiceLiveAgent represents a Chat configuration that is added to your web page. The EmbeddedServiceLiveAgent record contains a unique combination of a chat button and the Chat deployment that the administrator selects during setup.

To create an EmbeddedServiceLiveAgent record:

1. Create a Chat Deployment record.
2. Create a Chat Button record.
3. Create an EmbeddedServiceConfig record.
4. Set the fields for the Chat Deployment record, Chat Button record, and EmbeddedServiceConfig record as references on the EmbeddedServiceLiveAgent record.

Note: Any changes you make to the image fields override what you've entered in Setup. We recommend setting your image URLs in Setup.

Wildcard Support in the Manifest File

This metadata type doesn't support the wildcard character `*` (asterisk) in the `package.xml` manifest file. For information about using the manifest file, see Deploying and Retrieving Metadata with the Zip File.

EmbeddedServiceMenuSettings

Represents a setup node for creating a channel menu deployment. Channel menus list the ways in which customers can contact your business. This type extends the Metadata metadata type and inherits its `fullName` field.

File Suffix and Directory Location

EmbeddedServiceMenuSettings components are stored in the `developer_name.EmbeddedServiceMenuSettings` folder.

Version

EmbeddedServiceMenuSettings components are available in API version 47.0 and later.

Fields

Field Name	Field Type	Description
`branding`	string	The developer name of the associated BrandingSet.
`embeddedServiceCustomLabels`	EmbeddedServiceCustomLabel[]	Represents a customized label that appears in the embedded component for a particular channel menu deployment.
`embeddedServiceMenuItems`	EmbeddedServiceMenuItem[]	Represents a channel menu item that lists a way in which customers can contact your business.
`isEnabled`	boolean	If `true` (default), the deployment is enabled. If `false`, the deployment is disabled.
`masterLabel`	string	Required. The name of the channel menu deployment.
`site`	string	Required. The name of the Salesforce Community or site connected to this channel menu deployment.

EmbeddedServiceCustomLabel

Represents the custom labels used in your channel menu deployment.

Field Name	Field Type	Description
`customLabel`	string	The customized label that appears in the channel menu.

Field Name	Field Type	Description
`feature`	EmbeddedServiceFeature (enumeration of type string)	The feature using the custom label. For channel menu deployments, the value is `ChannelMenu`.
`labelKey`	EmbeddedServiceLabelKey (enumeration of type string)	The type of custom label. Channel menu label key values are as follows. The first two values apply to custom channel menu labels, and the remaining five values apply to menu item custom labels. • `CM_Container_Header_Primary_Greeting` • `CM_Container_Header_Secondary_Greeting` • `CM_Container_MenuItems_WebChatAvailable` • `CM_Container_MenuItems_WebChatUnavailable` • `CM_Container_MenuItems_WebChatLoading` • `CM_Container_MenuItems_ChannelLabel` • `CM_Container_Button_AssistiveText`

EmbeddedServiceMenuItem

Represents an item in a channel menu.

Field Name	Field Type	Description
`channel`	string	The ID of the channel type. If `channelType` is `Phone` or `CustomURL`, this field is `null`.
`channelType`	EmbeddedServiceChannelType (enumeration of type string)	The type of communication channel. Values are: • `EmbeddedServiceConfig` • `MessagingChannel` • `Phone` • `CustomURL`
`customUrl`	string	A custom URL that appears in the menu. The `shouldOpenUrlInSameTab` field determines where the URL opens.
`displayOrder`	int	The item's order in the menu, such as 1 or 2.
`embeddedServiceCustomLabels`	EmbeddedServiceCustomLabel[]	Represents the custom labels used in your channel menu item.
`iconUrl`	string	The icon URL for the menu item. Icons can be used only for phone, SMS, custom URL, and chat menu items.
`itemName`	string	A unique custom name for the menu item, which is visible in the user interface.
`osOptionsHideInIOS`	boolean	If `true`, the menu item is hidden in iOS.

Field Name	Field Type	Description
osOptionsHideInLinuxOS	boolean	If true, the menu item is hidden in Linux OS.
osOptionsHideInMacOS	boolean	If true, the menu item is hidden in Mac OS.
osOptionsHideInOtherOS	boolean	If true, the menu item is hidden in any operating system other than iOS, Linux, Mac, and Windows.
osOptionsHideInWindowsOS	boolean	If true, the menu item is hidden in Windows OS.
phoneNumber	string	The phone number for menu items whose channelType is Phone.
shouldOpenUrlInSameTab	boolean	If the menu item's channelType is CustomURL, this field indicates whether the link opens in the same tab (true) or a new tab (false).

Declarative Metadata Sample Definition

The following is an example of an EmbeddedServiceMenuSettings component.

```
<?xml version="1.0" encoding="UTF-8"?>
<EmbeddedServiceMenuSettings xmlns="http://soap.sforce.com/2006/04/metadata">
    <embeddedServiceCustomLabels>

<customLabel>CM_Container_Header_Primary_Greeting_3MsRM0000004CB5_6181150</customLabel>
        <labelKey>CM_Container_Header_Primary_Greeting</labelKey>
    </embeddedServiceCustomLabels>
    <embeddedServiceCustomLabels>

<customLabel>CM_Container_Header_Secondary_Greeting_3MsRM0000004CB5_4637097</customLabel>

        <labelKey>CM_Container_Header_Secondary_Greeting</labelKey>
    </embeddedServiceCustomLabels>
    <embeddedServiceMenuItems>
        <channel>Chat</channel>
        <channelType>EmbeddedServiceConfig</channelType>
        <displayOrder>1</displayOrder>
        <embeddedServiceCustomLabels>

<customLabel>CM_Container_MenuItems_WebChatUnavailable_3miRM0000004CuZ_8003848</customLabel>

            <labelKey>CM_Container_MenuItems_WebChatUnavailable</labelKey>
        </embeddedServiceCustomLabels>
        <embeddedServiceCustomLabels>

<customLabel>CM_Container_MenuItems_WebChatAvailable_3miRM0000004CuZ_5823055</customLabel>

            <labelKey>CM_Container_MenuItems_WebChatAvailable</labelKey>
        </embeddedServiceCustomLabels>
        <itemName>Chat1</itemName>
        <osOptionsHideInIOS>false</osOptionsHideInIOS>
        <osOptionsHideInLinuxOS>true</osOptionsHideInLinuxOS>
        <osOptionsHideInMacOS>false</osOptionsHideInMacOS>
        <osOptionsHideInOtherOS>false</osOptionsHideInOtherOS>
        <osOptionsHideInWindowsOS>true</osOptionsHideInWindowsOS>
        <shouldOpenUrlInSameTab>false</shouldOpenUrlInSameTab>
    </embeddedServiceMenuItems>
    <embeddedServiceMenuItems>
        <channelType>Phone</channelType>
        <displayOrder>2</displayOrder>
        <itemName>Phone1</itemName>
        <osOptionsHideInIOS>true</osOptionsHideInIOS>
        <osOptionsHideInLinuxOS>false</osOptionsHideInLinuxOS>
        <osOptionsHideInMacOS>true</osOptionsHideInMacOS>
        <osOptionsHideInOtherOS>false</osOptionsHideInOtherOS>
        <osOptionsHideInWindowsOS>false</osOptionsHideInWindowsOS>
        <phoneNumber>1234567890</phoneNumber>
        <shouldOpenUrlInSameTab>false</shouldOpenUrlInSameTab>
    </embeddedServiceMenuItems>
    <embeddedServiceMenuItems>
        <channelType>CustomURL</channelType>
        <customUrl>https://google.com</customUrl>
        <displayOrder>3</displayOrder>
        <itemName>url1</itemName>
        <osOptionsHideInIOS>false</osOptionsHideInIOS>
        <osOptionsHideInLinuxOS>false</osOptionsHideInLinuxOS>
        <osOptionsHideInMacOS>false</osOptionsHideInMacOS>
        <osOptionsHideInOtherOS>false</osOptionsHideInOtherOS>
        <osOptionsHideInWindowsOS>false</osOptionsHideInWindowsOS>
        <shouldOpenUrlInSameTab>false</shouldOpenUrlInSameTab>
    </embeddedServiceMenuItems>
    <isEnabled>true</isEnabled>
    <masterLabel>ChannelMenuSettings</masterLabel>
    <site>SnapInCommunity</site>
</EmbeddedServiceMenuSettings>
```

Wildcard Support in the Manifest File

This metadata type doesn't support the wildcard character `*` (asterisk) in the `package.xml` manifest file. For information about using the manifest file, see Deploying and Retrieving Metadata with the Zip File.

EntitlementProcess

Represents the settings for an entitlement process.

This type extends the Metadata metadata type and inherits its `fullName` field.

File Suffix and Directory Location

Entitlement process values are stored in files in the `entitlementProcesses` directory. Each file has the name of a process and the suffix `.entitlementProcess`. Each file contains one entitlement process or, if entitlement versioning is enabled, one version of an entitlement process.

The name of the file is the name of the entitlement process with the version appended to the end, if applicable (for example, an entitlement process named "gold_support" might have the file name "gold_support_v2.entitlementProcess"). This file name corresponds to the `slaProcess.NameNorm` field exposed through the SOAP API. This file name is distinct from the `name` field, which represents what displays in the user interface and, if versioning is enabled, might be shared among multiple versions of the same entitlement process. The `slaProcess.NameNorm` field contains the lowercase version of the `name` field shown in the user interface.

Version

Entitlement processes are available in API version 27.0 and later.

Fields

Field Name	Field Type	Description
`active`	boolean	Indicates whether the entitlement process is active (`true`) or not (`false`).
`businessHours`	string	The business hours that apply to the entitlement process. This field is available in API version 30.0 and later.
`description`	string	The description of the entitlement process.
`entryStartDateField`	string	For milestone processes on which a case enters the process based on a custom date/time field on the case, specifies which date and time are used. Valid values are: • SlaStartDate (entitlement process start date) • CreatedDate (date case was opened) • ClosedDate (date case was closed) • LastModifiedDate (date case was last modified) • StopStartDate (date case was stopped)
`exitCriteriaBooleanFilter`	string	For milestone processes on which a case exits the process when custom criteria are met, and for which filter logic is added, specifies that logic.
`exitCriteriaFilterItems`	FilterItem[]	For milestone processes on which a case exits the process when custom criteria are met, specifies those criteria.
`exitCriteriaFormula`	string	For milestone processes on which a case exits the process when a custom formula evaluates to true, specifies that formula.
`isVersionDefault`	boolean	Indicates whether the entitlement process is the default version (`true`) or not (`false`). This field is available in API version 28.0 and later.
`milestones`	EntitlementProcessMilestoneItem[]	Represents a milestone on the entitlement process.

Field Name	Field Type	Description
`name`	string	The name of the entitlement process as it displays in the user interface.
`SObjectType`	string	Indicates the type of record that the entitlement process can run on.
`versionMaster`	string	Identifies the sequence of versions to which this entitlement process belongs. This field's contents can be any value as long as it is identical among all versions of the entitlement process. This field is available in API version 28.0 and later.
`versionNotes`	string	The description of the entitlement process version. This field is available in API version 28.0 and later.
`versionNumber`	int	The version number of the entitlement process. Must be 1 or greater. This field is available in API version 28.0 and later.

EntitlementProcessMilestoneItem

Represents a milestone item on an entitlement process.

Fields

Field Name	Field Type	Description
`businessHours`	string	The business hours that apply to the milestone. This field is available in API version 30.0 and later.
`criteriaBooleanFilter`	string	For milestones that apply only when criteria are met and for which filter logic is added, specifies that logic.
`milestoneCriteriaFilterItems`	FilterItem[]	For milestones that apply only when criteria are met, specifies those criteria.
`milestoneCriteriaFormula`	string	For milestones that apply only when a formula evaluates to true, specifies that formula.
`milestoneName`	string	The name of the milestone.
`minutesCustomClass`	string	The name of the Apex class that is used to calculate the trigger time. This field is available in API version 30.0 and later.
`minutesToComplete`	int	The number of minutes from when the case enters the entitlement process that the milestone occurs.

Field Name	Field Type	Description
`successActions`	WorkflowActionReference[]	The actions triggered when the milestone is completed.
`timeTriggers`	EntitlementProcessMilestoneTimeTrigger[]	The time triggers on an entitlement process milestone.
`useCriteriaStartTime`	boolean	When the milestone starts: when the milestone criteria are met (true) or when the case enters the entitlement process (false).

EntitlementProcessMilestoneTimeTrigger

Represents the time trigger on an entitlement process milestone.

Fields

Field Name	Field Type	Description
`actions`	WorkflowActionReference[]	The actions to take when the time trigger is reached, if, at that time, the milestone is not completed.
`timeLength`	int	The length of time between the time trigger activation and the milestone target completion date. This may be a negative or positive value. Negative values indicate that the target completion date has not yet arrived and correspond to warning time triggers. Positive values indicate that the target completion date has passed and correspond to violation time triggers.
`workflowTimeTriggerUnit`	MilestoneTimeUnits (enumeration of type string)	Specifies the type of unit used to determine when a workflow should be triggered. Valid values are: • `Minutes` • `Hours` • `Days`

Declarative Metadata Sample Definition

This is a sample entitlement process.

```
<?xml version="1.0" encoding="UTF-8"?>
<EntitlementProcess xmlns="http://soap.sforce.com/2006/04/metadata">
    <active>true</active>
    <description>eppersone</description>
    <entryStartDateField>SlaStartDate</entryStartDateField>
    <exitCriteriaBooleanFilter>1 OR 2</exitCriteriaBooleanFilter>
    <exitCriteriaFilterItems>
        <field>Case.IsClosed</field>
        <operation>equals</operation>
        <value>true</value>
    </exitCriteriaFilterItems>
    <exitCriteriaFilterItems>
        <field>Case.Description</field>
        <operation>startsWith</operation>
        <value>foo</value>
    </exitCriteriaFilterItems>
    <milestones>
        <milestoneName>m1</milestoneName>
        <minutesToComplete>1</minutesToComplete>
        <successActions>
            <name>emailBob</name>
            <type>Alert</type>
        </successActions>
        <timeTriggers>
            <actions>
                <name>emailAlice</name>
                <type>Alert</type>
            </actions>
            <actions>
                <name>setEscalateToTrue</name>
                <type>FieldUpdate</type>
            </actions>
            <timeLength>1</timeLength>
            <workflowTimeTriggerUnit>Minutes</workflowTimeTriggerUnit>
        </timeTriggers>
        <timeTriggers>
            <actions>
                <name>setStopToTrue</name>
                <type>FieldUpdate</type>
            </actions>
            <timeLength>2</timeLength>
            <workflowTimeTriggerUnit>Minutes</workflowTimeTriggerUnit>
        </timeTriggers>
        <useCriteriaStartTime>false</useCriteriaStartTime>
    </milestones>
    <milestones>
        <milestoneCriteriaFilterItems>
            <field>Case.Priority</field>
            <operation>equals</operation>
            <value>High</value>
        </milestoneCriteriaFilterItems>
        <milestoneName>m2</milestoneName>
        <minutesToComplete>120</minutesToComplete>
        <useCriteriaStartTime>true</useCriteriaStartTime>
        <successActions>
            <name>emailBob</name>
            <type>Alert</type>
        </successActions>
    </milestones>
</EntitlementProcess>
```

Wildcard Support in the Manifest File

This metadata type supports the wildcard character `*` (asterisk) in the `package.xml` manifest file. For information about using the manifest file, see Deploying and Retrieving Metadata with the Zip File.

EntitlementTemplate

Represents an entitlement template. Entitlement templates are predefined terms of customer support that you can quickly add to products. For example, you can create entitlement templates for Web or phone support so that users can easily add entitlements to products offered to customers.

EntitlementTemplate extends the Metadata metadata type and inherits its `fullName` field.

Declarative Metadata File Suffix and Directory Location

EntitlementTemplate components are stored in the `entitlementTemplates` directory of the corresponding package directory. The file name matches the unique name of the entitlement template, and the extension is `.entitlementTemplate`.

Version

Lightning Platform EntitlementTemplate components are available in API version 18.0 and higher.

Fields

Field	Field Type	Description
`businessHours`	string	The entitlement's supported business hours.
`casesPerEntitlement`	int	Lets you limit the number of cases the entitlement supports.
`entitlementProcess`	string	The entitlement process associated with the entitlement.
`isPerIncident`	boolean	`true` if entitlements created from this template service a limited number of cases; `false` otherwise.
`term`	int	The number of days the entitlement is in effect.
`type`	string	The type of entitlement, such as Web or phone support.

Declarative Metadata Sample Definition

A sample XML definition of an entitlement template is shown below.

```
<?xml version="1.0" encoding="UTF-8"?>
<EntitlementTemplate xmlns="http://soap.sforce.com/2006/04/metadata">
    <businessHours>AlternateBusinessHours</businessHours>
    <casesPerEntitlement>12</casesPerEntitlement>
    <entitlementProcess>Process1</entitlementProcess>
    <isPerIncident>true</isPerIncident>
    <term>33</term>
    <type>Phone Support</type>
</EntitlementTemplate>
```

Wildcard Support in the Manifest File

This metadata type supports the wildcard character * (asterisk) in the package.xml manifest file. For information about using the manifest file, see Deploying and Retrieving Metadata with the Zip File.

EscalationRules

Represents case escalation rules to escalate cases automatically if they are not resolved within a certain period of time. You can access rules metadata for all applicable objects, for a specific object, or for a specific rule on a specific object.

The package.xml syntax for accessing all escalation rules for all objects is:

```
    <types>
        <members>*</members>
        <name>EscalationRules</name>
    </types>
```

All rules for a specific object uses a similar syntax without the wildcard. For example, all escalation rules for the Case object would use this syntax:

```
    <types>
        <members>Case</members>
        <name>EscalationRules</name>
    </types>
```

You can also access specific escalation rules for an object. The following example only accesses the “samplerule” and “newrule” escalation rules on the Case object. Notice that for this example the type name syntax is EscalationRule and not EscalationRules.

```
    <types>
        <members>Case.samplerule</members>
        <members>Case.newrule</members>
        <name>EscalationRule</name>
    </types>
```

File Suffix and Directory Location

EscalationRules for an object have the suffix .escalationRules and are stored in the escalationRules folder. For example, all Case escalation rules are stored in the Case.escalationRules file.

Version

EscalationRules components are available in API version 27.0 and later.

Fields

Field Name	Field Type	Description
`escalationRule`	EscalationRule[]	Represents one escalation rule and specifies whether it is active or not. Escalation rules are processed in the order they appear in the EscalationRules container.

EscalationRule

Field Name	Field Type	Description
`active`	boolean	Indicates whether the escalation rule is active (`true`) or not (`false`).
`fullname`	string	Inherited from Metadata, this field is not defined in the WSDL for this metadata type. It must be specified when creating, updating, or deleting. See `createMetadata()` to see an example of this field specified for a call. This value cannot be `null`.
`ruleEntry`	`RuleEntry[]`	Contains the definitions of the rule entries in the escalation rule.

RuleEntry

Represents the fields used by the rule.

Field Name	Field Type	Description
`booleanFilter`	string	Advanced filter conditions that were specified for the rule.
`businessHours`	string	The hours at which escalation actions are performed. Specify only if `businessHoursSource` is set to `Static`.
`businessHoursSource`	`BusinessHoursSourceType` (enumeration of type string)	Valid values are: • `None` • `Case` • `Static`
`criteriaItems`	`FilterItem`	The items in the list that define the assignment criteria.
`disableEscalationWhenModified`	boolean	Indicates whether the escalation is disabled when the record is modified `true`) or not (`false`).
`escalationAction`	`EscalationAction[] on page 482`	The actions to perform when the escalation criteria are met.

Field Name	Field Type	Description
`escalationStartTime`	`EscalationStartTimeType` (enumeration of type string)	Indicates the start time for the escalation. Valid values are: • `CaseCreation` • `CaseLastModified`
`formula`	string	The validation formula. Note: Specify either `formula` or `criteriaItems`, but not both fields.

EscalationAction

Describes the action to take for an escalation rule.

Field Name	Field Type	Description
`assignedTo`	string	The name of the user or queue the item is assigned to.
`assignedToTemplate`	string	Specifies the template to use for the email that is automatically sent to the new owner specified by the escalation rule. Lightning email templates aren't packageable. We recommend using a Classic email template.
`assignedToType`	`AssignToLookupValueType` (enumeration of type string)	Valid values are: • `User` • `Queue`
`minutesToEscalation`	int	The number of minutes until the escalation occurs.
`notifyCaseOwner`	boolean	Indicates that the owner of the case is notified when the case is escalated `true`) or not (`false`).
`notifyEmail`	string	Specifies the email address of the user to notify.
`notifyTo`	string	Specifies the user to notify.
`notifyToTemplate`	string	Specifies the template to user for the notification email.

Declarative Metadata Sample Definition

The following is an example EscalationRules component:

```
<EscalationRules xmlns="http://soap.sforce.com/2006/04/metadata">
    <escalationRule>
        <fullName>samplerule</fullName>
        <active>false</active>
        <ruleEntry>
            <businessHours>test</businessHours>
            <businessHoursSource>Static</businessHoursSource>
            <criteriaItems>
                <field>Case.Description</field>
                <operation>contains</operation>
                <value>test</value>
            </criteriaItems>
            <escalationAction>
                <assignedTo>someuser@org.com</assignedTo>
                <assignedToTemplate>emailtemplatename</assignedToTemplate>
                <assignedToType>User</assignedToType>
                <minutesToEscalation>1440</minutesToEscalation>
                <notifyCaseOwner>false</notifyCaseOwner>
            </escalationAction>
            <escalationStartTime>CaseLastModified</escalationStartTime>
        </ruleEntry>
    </escalationRule>
</EscalationRules>
```

EventDelivery

Represents how an event instance maps to a target payload. Removed in API version 46.0. This type extends the Metadata metadata type and inherits its `fullName` field.

File Suffix and Directory Location

Event delivery components have the suffix file path `.delivery`, and are stored in the `eventDeliveries` folder.

Version

Event delivery components are available in API versions 41.0 to 45.0.

Limits

Your org can have a maximum of 2500 EventDelivery object instances.

Fields

Field Name	Field Type	Description
eventParameters	EventParameterMap[]	An array of parameters to deliver in addition to the published event's data.
eventSubscription	string	Required. The ID of the subscription to deliver the data to.
referenceData	string	User-defined non-unique identifier.

Field Name	Field Type	Description
type	EventDeliveryType (enumeration of type string)	Required. Determines what action occurs when the event is delivered to the listeners on behalf of the subscribers. Valid values are: • StartFlow—When the event occurs, it's delivered to a flow of type CustomEvent. Those flows are built through Process Builder. • ResumeFlow—Reserved for future use.

EventParameterMap

Parameters to deliver in addition to the published event's data.

If type is StartFlow, you must include a parameter where parameterName is FlowVersionName and parameterValue is the name of the flow that you want to start. The flow name must include its version number. For example, myFlow-3.

Each event delivery can have up to 10 parameters.

Field Name	Field Type	Description
parameterName	string	The parameter name.
parameterValue	string	The parameter value.

Declarative Metadata Sample Definition

The following is an example of an event delivery file.

```
<?xml version="1.0" encoding="UTF-8"?>
<EventDelivery xmlns="http://soap.sforce.com/2006/04/metadata">
   <eventParameters>
      <parameterName>FlowVersionName</parameterName>
      <parameterValue>My_Event_Based_Process-1</parameterValue>
   </eventParameters>
   <eventSubscription>MySubscription</eventSubscription>
   <referenceData>My_Event_Based_Process_1</referenceData>
   <type>StartFlow</type>
</EventDelivery>
```

The following is an example package.xml that deploys or retrieves all the available event delivery metadata in your org.

```
<?xml version="1.0" encoding="UTF-8"?>
<Package xmlns="http://soap.sforce.com/2006/04/metadata">
   <types>
      <members>EventDelivery</members>
      <name>*</name>
   </types>
</Package>
```

Wildcard Support in the Manifest File

This metadata type supports the wildcard character `*` (asterisk) in the `package.xml` manifest file. For information about using the manifest file, see Deploying and Retrieving Metadata with the Zip File.

EventSubscription

Represents a subscription to an event type. Removed in API version 46.0. This type extends the Metadata metadata type and inherits its `fullName` field.

File Suffix and Directory Location

EventSubscription components have the suffix file path `.subscription`, and are stored in the `eventSubscriptions` folder.

Version

Event subscription components are available in API versions 41.0 to 45.0.

Limits

Your org can have a maximum of:

- 4,000 total event subscriptions
- 2,000 active event subscriptions

Fields

Field Name	Field Type	Description
`active`	boolean	If the subscription isn't active, it never receives any events.
`eventParameters`	EventParameterMap[]	An array of parameters that must be true for published events.
`eventType`	string	Required. The name of the platform event.
`referenceData`	string	Required. If the subscriber is a flow of type CustomEvent, `referenceData` is ***`flowName_versionNumber`***. For example, `Printer_Management_2`.

EventParameterMap

An array of parameters that must be true for published events. For example, subscribe to Vendor Response events only if `Status__c` is `Shipped`.

Each event subscription can have up to 10 parameters.

Field Name	Field Type	Description
`parameterName`	string	Required. The published event's field name.

Field Name	Field Type	Description
parameterValue	string	The value that must be true.

Declarative Metadata Sample Definition

The following is an example of an active event subscription.

```
<?xml version="1.0" encoding="UTF-8"?>
<EventSubscription xmlns="http://soap.sforce.com/2006/04/metadata">
    <active>true</active>
    <eventType>Printer_Status__e</eventType>
    <referenceData>Printer_Management</referenceData>
</EventSubscription>
```

The following is an example of an inactive event subscription that sets event parameters.

```
<?xml version="1.0" encoding="UTF-8"?>
<EventSubscription xmlns="http://soap.sforce.com/2006/04/metadata">
   <name>MySubscription</name>
   <active>false</active>
   <eventParameters>
      <parameterName>Ink_Status__c</parameterName>
      <parameterValue>low</parameterValue>
   </eventParameters>
   <eventParameters>
      <parameterName>Serial_Number__c</parameterName>
      <parameterValue>00123456789</parameterValue>
   </eventParameters>
   <eventType>Printer_Status__e</eventType>
   <referenceData>My_Event_Based_Process_1</referenceData>
</EventSubscription>
```

The following is an example `package.xml` that deploys or retrieves all the available event subscription metadata in your org.

```
<?xml version="1.0" encoding="UTF-8"?>
<Package xmlns="http://soap.sforce.com/2006/04/metadata">
   <types>
      <members>*</members>
      <name>EventSubscription</name>
   </types>
   <version>41.0</version>
</Package>
```

Wildcard Support in the Manifest File

This metadata type supports the wildcard character `*` (asterisk) in the `package.xml` manifest file. For information about using the manifest file, see Deploying and Retrieving Metadata with the Zip File.

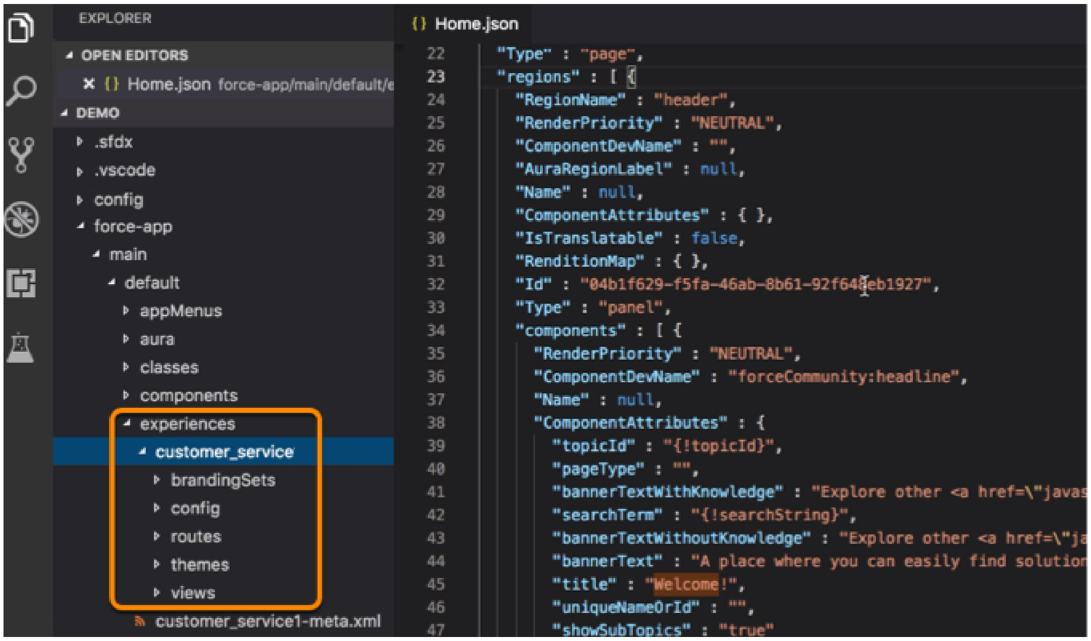
ExperienceBundle

Represents a text-based code structure of community settings and site components, such as pages, branding sets, and themes. Developers can quickly update and deploy one or more Lightning communities *programmatically* using their preferred development tools.This type extends the Metadata metadata type and inherits its `fullName` field.

File Suffix and Directory Location

ExperienceBundle components have the suffix `.json` and are stored in the `experiences` folder when retrieved. Each community in your org has its own folder. Each of these folders contains other folders for the supported properties.

The ExperienceBundle can contain one or more community definitions under the `experiences` folder. Each community definition has resource folders for brandingSets, config, routes, themes, variations, and views, each with additional, related configuration information in `JSON` files. Here's an example community definition, showing the resource folders.



```
"Type" : "page",
"regions" : [ {
  "RegionName" : "header",
  "RenderPriority" : "NEUTRAL",
  "ComponentDevName" : "",
  "AuraRegionLabel" : null,
  "Name" : null,
  "ComponentAttributes" : { },
  "IsTranslatable" : false,
  "RenditionMap" : { },
  "Id" : "04b1f629-f5fa-46ab-8b61-92f648eb1927",
  "Type" : "panel",
  "components" : [ {
    "RenderPriority" : "NEUTRAL",
    "ComponentDevName" : "forceCommunity:headline",
    "Name" : null,
    "ComponentAttributes" : {
      "topicId" : "{!topicId}",
      "pageType" : "",
      "bannerTextWithKnowledge" : "Explore other <a href=\"javas
      "searchTerm" : "{!searchString}",
      "bannerTextWithoutKnowledge" : "Explore other <a href=\"ja
      "bannerText" : "A place where you can easily find solution
      "title" : "Welcome!",
      "uniqueNameOrId" : "",
      "showSubTopics" : "true"
```

Version

ExperienceBundle components are available in API version 46.0 and later.

Special Access Rules

To use the ExperienceBundle metadata type, from Setup, enter *`Communities Settings`* in the Quick Find box, and then select **Communities Settings**. Select **Enable ExperienceBundle Metadata API**, and save your changes.

Fields

Field Name	Field Type	Description
experienceResources	ExperienceResources[]	The list of resources in this ExperienceBundle. Each resource represents an artifact of a community such as brandingSets, config, routes, themes, variations, and views.
label	string	Required. Represents the name of the ExperienceBundle.
type	SiteType (enumeration of type string)	Required. Identifies the kind of site. Only Lightning communities are supported, using the value ChatterNetworkPicasso.

ExperienceResources

Represents a list of communities in the bundle.

Field Name	Field Type	Description
experienceResource	ExperienceResource[]	The list of resources in this ExperienceBundle. Each resource represents a property for the community, such as brandingSets, config, routes, themes, and views.

ExperienceResource

Represents specific community information included in the ExperienceBundle.

Each type has a folder in the structure. Each folder contains one or more files providing information about that type and the site. Each corresponds to a specific folder and file in the ExperienceBundle.

Field Name	Field Type	Description
fileName	string	Required. Name of resource file.
format	string	Required. Only JSON is allowed.
source	base64	The JSON content of each file.
type	string	Required. The type of the resource. Valid values are: • brandingSets • config • routes • themes • views

Folders and Bundled Definitions

Each ExperienceBundle includes folders and associated data that is contained in `JSON` files.

brandingSets Folder

This folder contains one `JSON` file per branding set, named *`brandingSets_name`*`.json`. Each file has the same structure and properties.

`<`*`brandingSets_name`*`>.json`

Property	Type	Description
`definitionName`	string	Required. Represents the name for the branding set that is used in grouping branding sets under a theme. Defined as *`theme`*:branding-*`theme`*. For example, if the community theme is Stella, the `definitionName` would be `stella:branding-stella`. In addition, there are several standard templates that have unique naming: • Customer Account Portal uses `cpt:branding-cpt` • Customer Service uses `service:branding-service` • Help Center uses `helpCenter:branding-helpCenter` • Partner Central uses `prm:branding-prm` • Build Your Own uses `starter:branding-starter` Note: The combination of `definitionName` + `label` must be unique in your org.
`id`	UUID	Represents the component's GUID.
`label`	string	Represents the name of the branding set. Note: The combination of `definitionName` + `label` must be unique in your org.
`type`	string	Represents the component type. The only supported value is `brandingSet`.

Property	Type	Description
`values`	map	Required. Represents a map of branding values that can be applied to a community.

```
{
  "values" : {
    "HeaderBackgroundColor" : "#FFFFFF",
    "TextTransformStyle" : "none",
    "BorderColor" : "#D4D4D4",
    "DetailTextColor" : "#5A5A5A",
    "HeaderFonts" : "Ek Mukta",
    "CardBackgroundColor" : "rgba(255, 255, 255, 0)",
    "LoginBackgroundColor" : "#F4F4F4",
    "_ActionColorTrans" : "rgba(25, 124, 190, 0.9)",
    "LoginBackgroundImage" :
"../../../../sfsites/picasso/core/external/salesforceIdentity/images/background.jpg?v=1",

    "PageBackgroundColor" : "#F5F7FA",
    "_HeaderTextColor" : "rgba(34,34,34,.8)",
    "_NavigationMenuHoverColor" : "rgba(255,255,255,.2)",
    "_HeaderInputBackgroundColor" : "rgba(255,255,255,.4)",
    "TextColor" : "#222222",
    "NavigationMenuTextColor" : "#222222",
    "_HeaderPlaceholderTextColor" : "rgba(85,85,85,.8)",
    "_OverlayTextColorShadow" : "#000000",
    "ActionColor" : "#0099DE",
    "CompanyLogo" : "",
    "_LinkColorDarker" : "#135F90",
    "_ActionColorDarker" : "#135F90",
    "_HoverColor" : "rgba(25, 124, 190, 0.05)",
    "ErrorFontColor" : "#ff9e9e",
    "OverlayTextColor" : "#FFFFFF",
    "PrimaryFont" : "Ek Mukta",
    "LinkColor" : "#3558D6"
    },
  "definitionName" : "cpt:branding-cpt",
  "label" : "Customer Account Portal",
  "id" : "283407c3-5938-4a6b-b97f-621cda6968c8",
  "type" : "brandingSet"
 }
```

config Folder

The `config` folder contains several `JSON` files.

- *`sitename`*`.json`
- `languages.json`
- *`page_name`*`.json`

Note: One for each single-page application in the site: `loginAppPage.json` and `mainAppPage.json`

sitename.json File Properties

Property	Type	Description
`forgotPasswordRouteId`	UUID	Represents the ID of the route to use when a user forgets their password. Note: Not supported if the active template for the community doesn't support login (such as Help Center).
`isAvailableToGuests`	boolean	Indicates whether public users have access to the site (`true`) or not (`false`). The default value is `false`. Note: Not supported if the active template for the community doesn't support login (such as Help Center).
`isFilteredComponentsView`	boolean	Indicates whether the list of components is filtered based on the current page type (`true`) or not (`false`). Some components require specific parameters from the page and don't work unless you manually configure them. The default value is `false`.
`isProgressiveRenderingEnabled`	boolean	Indicates whether the display order of page components is prioritized (`true`) or not (`false`). The default value is `false`.
`loginAppPageId`	UUID	Represents the ID of the login page. Note: Not supported if the active template for the community doesn't support login (such as Help Center).
`mainAppPageId`	UUID	Required. Represents the ID of the main page.
`preferredDomain`	string	Represents the name of the domain to use for indexing a community's pages. Improves search engine results. Available in API version 48.0 and later.
`preferredDomainId`	string	Represents the domain to use for indexing a community's pages. Improves search engine results. Removed in API version 48.0. Use `preferredDomainId` instead.

Property	Type	Description
`selfRegistrationRouteId`	UUID	Represents the ID of the login route to use for self-registration. Note: Not supported if the active template for the community doesn't support login (such as Help Center).
`type`	string	Represents the component type. The only supported value is `site`.

trustedSitesForScript container

When implemented, there is one `trustedSitesForScript` container in *`sitename`*`.json`.

Property	Type	Description
`id`	UUID	Represents the component's GUID.
`isActive`	boolean	Indicates if whitelisted item is active (`true`) and should be respected or inactive (`false`) and should not be treated as a whitelisted source. Default is `false`.
`trustedSiteName`	string	Name of the whitelisted source as it appears in the UI.
`trustedSiteUrl`	string	The fully qualified URL of the whitelisted source.
`type`	string	Represents the component type. The only supported value is `trustedSitesForScripts`.

```
{
   "isAvailableToGuests" : false,
   "isFilteredComponentsView" : false,
   "mainAppPageId" : "df9907cb-6e68-4ca1-8bb2-51173ca5374e",
   "loginAppPageId" : "58e9939a-84b2-498d-bbc5-7a89d89087fa",
   "selfRegistrationRouteId" : "ad5c8bf1-297f-4ad3-b47c-0e35d85f10ef",
   "forgotPasswordRouteId" : "e3139f6f-44d8-4eec-be9d-3609ce063039",
   "isProgressiveRenderingEnabled" : false,
   "preferredDomain" : "none",
   "selfRegistrationRouteId" : "b8fe8ab1-f266-41e1-a63b-4791165f3c1d",
   "trustedSitesForScript" : [ {
     "id" : "92c489e2-0b7b-4a48-9c88-bef7e8fe6f1b",
     "isActive" : true,
     "trustedSiteName" : "test",
     "trustedSiteUrl" : "https://123.com",
     "type" : "trustedSitesForScripts"
   }, {
     "id" : "92c489e2-0b7b-4a48-9c88-bef7e8fe6f1c",
     "isActive" : true,
     "trustedSiteName" : "test1",
     "trustedSiteUrl" : "https://1234.com",
     "type" : "trustedSitesForScripts"
  } ],
    "type" : "site"
}
```

`languages.json` **File Properties**

Property	Type	Description
`defaultCode`	string	Required. Represents the base language code plus the country code where used.
`defaultLabel`	string	Required. Defines the display label for the language.
`id`	UUID	Represents the component's GUID.
`type`	string	Represents the component type. The only supported value is `languageContainer`.

There is one section per supported language as a container in `languages.json`

language container

Property	Type	Description
`countryCode`	string	Represents the country code of the selected language. This string can be empty. It applies only when the selected language has variations depending on the country, like Arabic (Algeria) and Arabic (Bahrain). In this case, use `countryCode` to distinguish between them. For example:{ `languageCode" : "ar", "CountryCode" : "DZ", "Label" : "Arabic (Algeria) (DZ)",}, { "Code" : "ar", "CountryCode" : "BH", "Label" : "Arabic (Bahrain) (BH)",}`
`fallbackLanguageId`	UUID	Represents the language to use when no content is available for the selected language. For example, imagine that a community visitor chooses **Japanese** from the language selector, but there is no

Property	Type	Description
		content for that page in Japanese. Content is displayed in the fallback language.
`id`	UUID	Represents the component's GUID.
`isActive`	boolean	Indicates whether a language is available to community visitors in the language selector (`true`) or not (`false`). The default value is `true`.
`label`	string	Defines the display label for a language. The display label appears in any language selector components that you add to your community and in the language selector in Experience Builder.
`languageCode`	string	Represents the language code for the selected language.
`type`	string	Represents the component type. The only supported value is `language`.

```
{
    "defaultCode" : "en_US",
    "defaultLabel" : "English (US)",
    "id" : "04597c83-0b9d-4f16-9f4d-4ec28bd553b4",
    "type" : "languageContainer",
    "languages" : [ {
        "languageCode" : "af",
        "countryCode" : "",
        "isActive" : true,
        "label" : "Afrikaans",
        "fallbackLanguageId" : "c6e7fe67-55e0-47b3-ad58-bf49539249f0",
        "id" : "22036d6f-11ce-4f7b-b7f0-f2c409f817ea",
        "type" : "language"
        }
     ]
 }
```

The page file represents single-page applications in the community. One file per page, named `*page_name*.json`.

Note: Each Lightning community is actually a single-page application, which is a web app that loads a single `HTML` page. Single-page applications use multiple views to update the page dynamically as the user interacts with it.

`*page_name*.json` File Properties

Property	Type	Description
`cmsSettings`	map	Settings for the CMS Connect header and footer. Valid values: • `headerName`

Property	Type	Description
		- `headerUrl` - `headerPersonalization` - `footerName` - `footerUrl` - `footerPersonalization` Both source and target org must have the CMSConnect and CMSPersonalization org perms enabled for settings to be retrieved.
`currentThemeId`	UUID	Required. Represents the UUID of the site's current theme. This field is available for `mainAppPage.json` and `loginAppPage.json` (where applicable).
`headMarkup`	string	Required. Allows the addition of custom markup to the community's main page `<head>` tag. Similar to using **Experience Builder** > **Setting** > **Advanced** > **Head Markup** See Salesforce Help for markup guidance.
`id`	UUID	Required. Represents the component's GUID.
`isRelaxedCSPLevel`	boolean	Controls the ability to run scripts and script access to third-party hosts. The default is `false`. This field is available for `mainAppPage.json` and `loginAppPage.json` (where applicable).
`label`	string	Required. Represents the name of the page.
`templateName`	string	Required. Represents the unique developer name of the template page. Allowed values are: - Help Center - Customer Account Portal - Community Template - Customer Service - Build Your Own - Partner Central - *`Custom template`* (name which was used during the template creation)

Property	Type	Description
`type`	string	Required. Represents the component type. The only supported value is `appPage`.

```
{
    "headMarkup" : null,
    "isRelaxedCSPLevel" : false,
    "templateName" : "CPT Community Template",
    "cmsSettings" : { },
    "currentThemeId" : "ff52089c-6ad9-4dd9-b5b5-251d4a117ce3",
    "label" : "main",
    "id" : "df9907cb-6e68-4ca1-8bb2-51173ca5374e",
    "type" : "appPage"
}
```

routes Folder

The `routes` folder contains one `JSON` file per page, named `<page_name>.json`.

`<page_name>.json`

Property	Type	Description
`activeViewId`	UUID	Required. Represents the default view of the route. Used when there are no defined audiences or the user doesn't match any audience. Available in API version 48.0 and later.
`appPageId`	UUID	Required. Represents the Single Page Application (SPA) page for the route. It points to either `main.json` or `login.json`.
`id`	UUID	Required. Represents the component GUID. Inherited from the component.
`label`	string	Required. Represents the name of the route. Inherited from the component.
`objectApiName`	string	Required. The name of the custom object API. (Not available for standard objects.)
`pageAccess`	string	Required. Identifies the status of a route as public or private. When set to the default value `UseParent`, the status of the site determines the status of the route. Not editable from the user interface for routes that are always private. Valid values are

Property	Type	Description
		UseParent, Public, and RequireLogin.
routeType	string	Required. Identifies the type of route. Value is unique among all routes that share the same SPA page. The value in viewType must match.
type	string	Required. Represents the component type. The only supported value is route.
urlPrefix	string	Required. Represents the base URL for the route.

```
{
    "urlPrefix" : "",
    "appPageId" : "b5fe94e2-071f-47b2-b76d-427a624cb407",
    "routeType" : "home",
    "pageAccess" : "UseParent",
    "label" : "Home",
    "id" : "c7263124-7bc4-4147-a39a-25fe7e305b98",
    "type" : "route"
}
```

themes Folder

The themes folder contains one JSON file per theme named *theme_name*.json.

theme_name.json

Property	Type	Description
activeBrandingSetId	UUID	The id of the branding set currently in use. The branding set's definitionName must match the theme's brandingSetReference.
customCSS	string	Custom CSS for pages created from Lightning community template.
developerName	string	Required. The unique developer name of the theme. Most themes derive their names directly, for example Jepson uses jespon for its developerName. Standard templates have unique values: • cpt for Customer Account Portal • service for Customer Service • helpCenter for Help Center

Property	Type	Description
		• `prm` for Partner Central • `starter` for Build Your Own
`id`	UUID	Required. Represents the component's GUID.
`label`	string	Represents the name of the theme.
`layouts`	map	Required. Maps `ThemeLayoutType` to UUID, and contains the definition of the ThemeLayout. Login and Inner theme layouts are always required.
`type`	string	Required. Represents the component type. The only supported value is `theme`.

```
{
    "developerName" : "cpt",
    "layouts" : {
        "Login" : "12162c3e-06ac-43a9-adc7-db36ae5140b0",
        "Inner" : "c09d58be-0622-4fc4-806a-ed34174929f9"
    },
    "customCSS" : "",
    "activeBrandingSetId" : "283407c3-5938-4a6b-b97f-621cda6968c8",
    "label" : "Customer Account Portal",
    "id" : "ff52089c-6ad9-4dd9-b5b5-251d4a117ce3",
    "type" : "theme",
    "views" : [ {
        "componentName" : "salesforceIdentity:loginBody2",
        "label" : "Login",
        "id" : "12162c3e-06ac-43a9-adc7-db36ae5140b0",
        "type" : "view",
        "regions" : [ {
            "regionName" : "header",
            "id" : "f8354922-11f2-495d-9d89-0a51943af2b0",
            "type" : "region",
            "components" : [ ]
        } ]
    } ]
}
```

Note: Views can be children of a theme. These children are structured the same as views in the views folder.

variations Folder

The `variations` folder contains one `JSON` file per experience variation. The file is named *`experienceVariation_name`*`.json`. The name of your `JSON` should be the same as the `developerName` of your variation to avoid issues when deploying a community more than once.

There are three distinct types of variations supported: branding, page, and component visibility. Each variation file has the same structure and properties. The different variations are indicated through the `componentVariant` container.

Experience variations let you change the default behavior of the Lightning community. For example, you might want the community to show a special page variation for the home page when a user meets certain audience criteria. To achieve this, create an audience and target that audience to your experience variation using `targetId` in the `componentVariant` container of the experience variation definition file.

experienceVariation_name.json

Property	Type	Description
`componentVariants`	list	Required. A list of component variants that belong to this experience variation. Note: Only one component variant per experience variation is allowed in API 47.0 and later.
`developerName`	string	Required. The unique developer name of the experience variation. This name is used in the `targetValue` field of a Personalization API target and can't be updated after it's set. Note: For more information, see Audience.
`id`	UUID	Required. Represents the GUID of the component.
`type`	string	Required. Represents the type of the component. The only supported value is `experienceVariation`.

When implemented, there is one container in each *experienceVariation_name*.json file describing the variation.

componentVariant container

Property	Type	Description
`id`	UUID	Required. Represents the GUID of the component.
`propertyOverrides`	map	Required. Defines the property overrides for the given `targetId`. For example, if the `targetId` is pointing to a theme, you can override the `defaultBrandingSet` property of the theme to use a different brandingset for this experience variation. Supported property overrides:

Property	Type	Description
		• `activeBrandingSetId` where `targetId` is a theme. • `activeViewId` where `targetId` is a route. • `isVisible` where `targetId` is a component. Note: Only one entry in the map is allowed in API 47.0 and later.
`targetId`	UUID	Required. The UUID of the component whose properties you are overriding. Must be the `id` of a theme, route, or component.
`type`	string	Required. Represents the type of the component. The only supported value is `experienceVariation`.

Example of an experience variation for a branding set

```
{
   "id": "64e93604-78fa-11e9-8f9e-2a86e4085a59",
   "developerName": "BrandingVariation",
   "type": "experienceVariation",
   "componentVariants": [{
       "id": "4bf0af78-8d73-11e9-bc42-526af7764f64",
       "type": "componentVariant",
       // Theme UUID
       "targetId": "c810858e-78fa-11e9-8f9e-2a86e4085a59",
       "propertyOverrides": {
            // Brandingset UUID
            "activeBrandingSetId": "be9f4760-78fa-11e9-8f9e-2a86e4085a59"
       }
   }]
}
```

Example of an experience variation for a page variation

```
{
   "id": "64e93604-78fa-11e9-8f9e-2a86e4085a59",
   "developerName": "PageVariation",
   "type": "experienceVariation",
   "componentVariants": [{
       "id": "4bf0af78-8d73-11e9-bc42-526af7764f64",
       "type": "componentVariant",
       // Route UUID
       "targetId": "c810858e-78fa-11e9-8f9e-2a86e4085a59",
       "propertyOverrides": {
           // View UUID
           "activeViewId": "be9f4760-78fa-11e9-8f9e-2a86e4085a59"
       }
```

```
    }]
}
```

Example of an experience variation for component visibility

```
{
    "id": "64e93604-78fa-11e9-8f9e-2a86e4085a59",
    "developerName": "ComponentVisibilityVariation",
    "type": "experienceVariation",
    "componentVariants": [{
        "id": "4bf0af78-8d73-11e9-bc42-526af7764f64",
        "type": "componentVariant",
        // Component UUID
        "targetId": "c810858e-78fa-11e9-8f9e-2a86e4085a59",
        "propertyOverrides": {
             "isVisible": true
        }
    }]
}
```

views Folder

The `views` folder contains several `JSON` files that each define a view. Each Lightning community is built from single-page applications, which are web apps that load a single HTML page. Single-page applications consist of multiple views that update the page dynamically as the user interacts with it.

A *view* is made up of *regions* that contain other regions or *components* in the rendered page for the user. Within the `views` folder there is one file per view, named `view_name.json`.

Note: Single-page applications in your community are defined in the page files of the `config` folder.

`view_name.json`

Property	Type	Description
appPageId	UUID	Required. Single Page Application (SPA) page ID of the view. It points to either `main.json` or `login.json`.
componentName	string	Required. The FQN of the layout component. The component must implement `forceCommunity:layout` or, for theme layouts, `forceCommunity:themeLayout`
id	UUID	Required. Represents the GUID of the component.
label	string	Required. The name that appears in **Experience Builder** > **Settings** > **Theme** > **Configure**.
themeLayoutType	string	Theme layout type of the view (exposed only for views).

Property	Type	Description
`type`	string	Required. Represents the type of the component. The only supported value is `view`.
`viewType`	string	Required. Matches `routeType` for the route.

There are one or more regions as a container in each `<view_name>.json`

region container

Property	Type	Description
`id`	UUID	Required. Represents the component GUID.
`regionLabel`	string	Specifies region labels for tabs. Note: This property is present only for tab regions that are children of a component.
`regionName`	string	Required. Matches the design attribute in the design file of the layout component.
`type`	string	Required. Represents the component type. The only supported value is `region`.

There are one or more components as a container in the region section of each `<view_name>.json`

component container

Property	Type	Description
`componentAttributes`	HashMap	Required. The design attribute values of the component.
`componentName`	string	Required. The FQN of the component. Only components that can be used in the component panel in Experience Builder can be used in this field.
`id`	UUID	Required. Represents the component GUID.
`renderPriority`	enums.priority	Sets priority value for progressive rendering of the component. Possible Values: `HIGHEST`, `HIGH`, `NEUTRAL` Note: Only evaluated if the community has progressive rendering turned on in **Experience Builder** > **Settings** > **Advanced**.

Property	Type	Description
`renditionMap`	HashMap	Map of different rendition keys to UUIDs of RenditionComponents.
`type`	string	Required. Represents the component type. The only supported value is `component`.

Each component can have a rendition container in each `<view_name>.json`

rendition container

Property	Type	Description
`id`	UUID	Required. Represents the component GUID.
`renditionValue`	map	Map of different variations of a component, such as different languages of text.
`type`	string	Required. Represents the component type. The only supported value is `renditionComponent`.

```
{
    "themeLayoutType" : "Inner",
    "viewType" : "account-management",
    "appPageId" : "df9907cb-6e68-4ca1-8bb2-51173ca5374e",
    "componentName" : "siteforce:sldsOneColLayout",
    "label" : "Account Management",
    "id" : "9ca8fa47-8e87-4915-a6f7-c2d8d37f3076",
    "type" : "view",
    "regions" : [ {
         "regionName" : "content",
         "id" : "969ada98-7d72-4e45-8a10-7db51fae247c",
         "type" : "region",
         "components" : [ {
            "componentName" : "forceCommunity:tabset",
            "componentAttributes" : {
                "tabsetConfig" :
[illegible]

                "regions" : ""
            },
            "renderPriority" : "NEUTRAL",
            "renditionMap" : { },
            "id" : "4711850e-ffdc-4375-a45e-f716bcdbbb1c",
            "type" : "component",
            "renditions" : [ {
               "renditionValue" : {
                  "LumenInstanceAttributes" : {
                  "richTextValue" : "<p>new text</p>"
                  }
               },
              "id" : "9d8878df-f520-4010-861c-57b930a3daab",
              "type" : "renditionComponent"
            } ]
        } ]
    } ]
}
```

Declarative Metadata Sample Definition

Here's an example of an ExperienceBundle declaration. For individual folder and file examples for the bundled code, see brandingSets, config, routes, themes, variations, and views.

```
<xsd:complexType name="ExperienceBundle">
    <xsd:complexContent>
        <xsd:extension base="tns:Metadata">
            <xsd:sequence>
                <xsd:element name="experienceResources" minOccurs="0"
type="tns:ExperienceResources"/>
                <xsd:element name="label" type="xsd:string"/>
                <xsd:element name="type" type="tns:SiteType"/>
            </xsd:sequence>
        </xsd:extension>
    </xsd:complexContent>
</xsd:complexType>
    <xsd:complexType name="ExperienceResources">
        <xsd:sequence>
            <xsd:element name="experienceResource" minOccurs="0" maxOccurs="unbounded"
type="tns:ExperienceResource"/>
        </xsd:sequence>
    </xsd:complexType>
<xsd:complexType name="ExperienceResource">
    <xsd:sequence>
        <xsd:element name="fileName" type="xsd:string"/>
        <xsd:element name="format" type="xsd:string"/>
        <xsd:element name="source" minOccurs="0" type="xsd:base64Binary"/>
        <xsd:element name="type" type="xsd:string"/>
    </xsd:sequence>
</xsd:complexType>
```

Wildcard Support in the Manifest File

This metadata type supports the wildcard character `*` (asterisk) in the `package.xml` manifest file. For information about using the manifest file, see Deploying and Retrieving Metadata with the Zip File.

SEE ALSO:

ExperienceBundleSettings

Developer Guide: ExperienceBundle for Lightning Communities

ExternalDataSource

Represents the metadata associated with an external data source. Create external data sources to manage connection details for integration with data and content that are stored outside your Salesforce org.

Note: All credentials stored within this entity are encrypted under a framework that is consistent with other encryption frameworks on the platform. Salesforce encrypts your credentials by auto-creating org-specific keys. Credentials encrypted using the previous encryption scheme have been migrated to the new framework.

This type extends the Metadata metadata type and inherits its `fullName` field.

File Suffix and Directory Location

ExternalDataSource components are stored in the `dataSources` directory of the corresponding package directory. ExternalDataSource components have the suffix `.dataSource`, and the prefix is the name of the external data source.

Version

ExternalDataSource components are available in API version 28.0 and later.

Fields

Field Name	Field Type	Description
`authProvider`	string	The authentication provider represented by the AuthProvider component.
`certificate`	string	If you specify a certificate, your Salesforce org supplies it when establishing each two-way SSL connection with the external system. The certificate is used for digital signatures, which verify that requests are coming from your Salesforce org.
`customConfiguration`	string	A string of configuration parameters that are specific to the external data source's `type`. • customConfiguration for Salesforce Connect—Cross-Org Adapter • customConfiguration for Salesforce Connect—OData 2.0 or 4.0 Adapter • customConfiguration for Salesforce Connect—Custom Adapter
`customHttpHeader`	CustomHttpHeader[]	Represents custom HTTP headers used with OData 2.0 or OData 4.0 connectors. Available in API version 43.0 or later.

Field Name	Field Type	Description
`endpoint`	string	The URL of the external system, or if that URL is defined in a named credential, the named credential URL. Corresponds to `URL` in the user interface. A named credential URL contains the scheme `callout:`, the name of the named credential, and an optional path. For example: `callout:`*`My_Named_Credential`*`/`*`some_path`*. You can append a query string to a named credential URL. Use a question mark (?) as the separator between the named credential URL and the query string. For example: `callout:`*`My_Named_Credential`*`/`*`some_path`*`?format=json`.
`isWritable`	boolean	Lets the Lightning platform and users in this org create, update, and delete records for external objects associated with the external data source. The external object data is stored outside the org. By default, external objects are read only. Corresponds to `Writable External Objects` in the user interface. Available in API version 35.0 and later. However, with the cross-org adapter for Salesforce Connect, you can set this field to `true` only in API version 39.0 and later.
`label`	string	A user-friendly name for the external data source. The label is displayed in the Salesforce user interface, such as in list views. Examples include Acme Team Marketing Site, or Acme SharePoint.
`oauthRefreshToken`	string	The OAuth refresh token. Used to obtain a new access token for an end user when a token expires.
`oauthScope`	string	Specifies the scope of permissions to request for the access token. Corresponds to `Scope` in the user interface.
`oauthToken`	string	The access token issued by the external system.
`password`	string	The password to be used by your org to access the external system. Ensure that the credentials you use have adequate privileges to access the external system, perform searches, return data, and return information about the external system's metadata.
`principalType`	`External PrincipalType` (enumeration of type string)	Determines whether you're using one set or multiple sets of credentials to access the external system. Corresponds to `Identity Type` in the user interface. The valid values are: • `Anonymous` • `PerUser` • `NamedUser`

Field Name	Field Type	Description
`protocol`	`AuthenticationProtocol` (enumeration of type string)	The authentication protocol that's required to access the external system. The valid values are: • `NoAuthentication` • `Oauth` • `Password` For cloud-based Files Connect external systems, select **Oauth 2.0**. For on-premises systems, select **Password Authentication**. For Simple URL data sources, select **No Authentication**.
`repository`	string	Used for SharePoint Online. If metadata is not accessible, use this field to create tables and default table fields.
`type`	`ExternalDataSourceType` (enumeration of type string)	For Salesforce Connect, specifies the adapter that connects to the external system. The valid values are: • `OData`—OData 2.0 adapter • `OData4`—OData 4.0 adapter • `SfdcOrg`—cross-org adapter • *`ApexClassId`*—`DataSource.Provider` class that defines the custom adapter created via the Apex Connector Framework For Files Connect, specifies the data source type. The valid values are: • `ContentHubSharepoint`—SharePoint 2010 or 2013 • `ContentHubSharepointOffice365`—SharePoint Online • `ContentHubSharepointOneDrive`—OneDrive for Business • `ContentHubGDrive`—Google Drive If Chatter is enabled, you can also specify `SimpleURL` to access data hosted on a web server that doesn't require authentication. • `outgoingemail`—A data source used for sending an email through a quick action. The `Identity` and `Wrapper` types are reserved for future use. For the federated search external data source type, the valid value is `OpenSearch`.
`username`	string	The username to be used by your org to access the external system. Ensure that the credentials you use have adequate privileges to access the external system, perform searches, return data, and return information about the external system's metadata.
`version`	string	Reserved for future use.

CustomHttpHeader

Represents a custom HTTP header used with OData 2.0 or OData 4.0 connectors. Available in API version 43.0 or later.

Field Name	Field Type	Description
`description`	string	Description of the custom HTTP header.
`headerFieldName`	string	Required. Name of the custom HTTP header field.
`headerFieldValue`	string	Required. Value of the custom HTTP header field derived from a formula.
`isActive`	boolean	Specifies whether the custom HTTP header field is active (`true`) or inactive (`false`).

`customConfiguration` for Salesforce Connect—Cross-Org Adapter

The following sample JSON-encoded configuration string defines parameters that apply when the external data source's `type` is set to `SfdcOrg`.

```
{"apiVersion":"32.0","environment":"CUSTOM",
"searchEnabled":"true","timeout":"120"}
```

The parameters correspond to these fields in the user interface:

- `apiVersion—API Version`
- `environment—Connect to`
- `searchEnabled—Enable Search`
- `timeout—Connection Timeout`

`customConfiguration` for Salesforce Connect—OData 2.0 or 4.0 Adapter

The following JSON-encoded configuration string defines parameters that apply when the external data source's `type` is set to `OData` or `OData4`.

```
{"inlineCountEnabled":"true","csrfTokenName":"X-CSRF-Token",
"requestCompression":"false","pagination":"CLIENT",
"noIdMapping":"false","format":"ATOM",
"searchFunc":"","compatibility":"DEFAULT",
"csrfTokenEnabled":"true","timeout":"120",
"searchEnabled":"true"}
```

The parameters correspond to these fields in the user interface.

- `compatibility—Special Compatibility`
- `csrfTokenEnabled—CSRF Protection`
- `csrfTokenName—Anti-CSRF Token Name`
- `format—Format`
- `inlineCountEnabled—Request Row Counts`
- `noIdMapping—High Data Volume`
- `pagination—Server Driven Pagination`

- requestCompression—Compress Requests
- searchEnabled—Enable Search
- searchFunc—Custom Query Option for Salesforce Search
- timeout—Connection Timeout

customConfiguration for Salesforce Connect—Custom Adapter

The following sample JSON-encoded configuration string defines the parameter that applies when the external data source's type is set to the ID of a DataSource.Provider class.

```
{"noIdMapping":"false"}
```

The noIdMapping parameter corresponds to the High Data Volume field in the user interface.

Declarative Metadata Sample Definition

The following is the definition of an external data source for Salesforce Connect—OData 2.0 or 4.0 adapter.

```
<?xml version="1.0" encoding="UTF-8"?>
<ExternalDataSource xmlns="http://soap.sforce.com/2006/04/metadata">
    <authProvider>FacebookAuth</authProvider>
    <customConfiguration>{"compatibility":"DEFAULT",
    "noIdMapping":"false","inlineCountEnabled":"true",
    "searchEnabled":"true","format":"ATOM",
    "requestCompression":"false","pagination":"SERVER",
    "timeout":"120"}</customConfiguration>
    <customHttpHeaders>
        <headerFieldName>X-User</headerFieldName>
        <headerFieldValue>$User.Username</headerFieldValue>
    </customHttpHeaders>
    <endpoint>http://myappname.herokuapp.com/DataHub.svc</endpoint>
    <label>DataHub</label>
    <principalType>NamedUser</principalType>
    <protocol>Oauth</protocol>
    <type>OData</type>
</ExternalDataSource>
```

Wildcard Support in the Manifest File

This metadata type supports the wildcard character * (asterisk) in the package.xml manifest file. For information about using the manifest file, see Deploying and Retrieving Metadata with the Zip File.

ExternalServiceRegistration

Represents the External Service configuration for an org. This type extends the Metadata metadata type and inherits its fullName field.

File Suffix and Directory Location

ExternalServiceRegistration components have the suffix `.externalServiceRegistration` and are stored in the `externalServiceRegistrations` folder.

Version

ExternalServiceRegistration components are available in API version 39.0 and later.

Note: In API version 47.0 and earlier, external services supported Interagent and a limited subset of OpenAPI 2.0. As of API version 48.0, external services supports OpenAPI 2.0 but does not support InteragentHyperSchema. See “Enhanced External Services Considerations” in Salesforce Help.

Fields

Field Name	Field Type	Description
`description`	string	Required. The external service description defined when the service is created.
`label`	string	The service name as it appears on the External Services wizard.
`namedCredential`	string	Required. The reference by name to be used for the service.
`schema`	string	Required. The content of the JSON schema in the OpenAPI 2.0 format. Nillable.
`schemaType`	string	The schema format. OpenAPI for Open API 2.0 or InteragentHyperSchema for API version 47.0 and earlier. If not specified, schema type is derived based on the schema content. Nillable.
`schemaUrl`	string	Required. The path should begin with "/" and be relative to the named credential endpoint.
`status`	string	Indicates whether the service registration is complete, valid, and ready to use (complete) or not (incomplete).

Declarative Metadata Sample Definition

The following is an example of an ExternalServiceRegistration component that references an external credit service.

```
<?xml version="1.0" encoding="UTF-8"?>
<ExternalServiceRegistration xmlns="http://soap.sforce.com/2006/04/metadata">
  <label>BankService</label>
  <namedCredential>Bank</namedCredential>
  <schema>{
  &quot;swagger&quot;: &quot;2.0&quot;,
  &quot;basePath&quot;: &quot;/&quot;,
  &quot;info&quot;: {
    &quot;version&quot;: &quot;1.0&quot;,
    &quot;title&quot;: &quot;External Service for demo bank&quot;,
    &quot;description&quot;: &quot;### External Service for demo bank&quot;,
    &quot;x-vcap-service-name&quot;: &quot;DemoBankRestServices&quot;
  },
  ...
  &quot;paths&quot;: {
  &quot;/accounts/{accountName}&quot;: {
      ...
    }
  },
  &quot;definitions&quot;: {
    &quot;accountDetails&quot;: {
      ...
    },
    &quot;errorModel&quot;: {
      ...
    }
  }
}</schema>
  <schemaType>OpenApi</schemaType>
  <schemaUrl>/accounts/schema</schemaUrl>
  <status>Complete</status>
</ExternalServiceRegistration>
```

Wildcard Support in the Manifest File

This metadata type supports the wildcard character `*` (asterisk) in the `package.xml` manifest file. For information about using the manifest file, see Deploying and Retrieving Metadata with the Zip File.

FeatureParameterBoolean

Represents a boolean feature parameter in a packaging org that has access to the Feature Management App (FMA). Feature parameters let you drive app behavior and track activation metrics in subscriber orgs that install your package. This type extends the Metadata metadata type and inherits its `fullName` field.

File Suffix and Directory Location

FeatureParameterBoolean components have the suffix `.featureParameterBoolean`. The components are stored in the `featureParameters` folder, which contains components for all the feature parameter metadata types.

Version

FeatureParameterBoolean components are available in API version 41.0 and later.

Special Access Rules

Available only in packaging orgs that have access to the Feature Management App (FMA). For details, see Manage Features in the *ISVforce Guide*.

Fields

Field Name	Field Type	Description
dataFlowDirection	FeatureParameterDataFlowDirection	After a package containing the components is installed, indicates whether the feature parameter's value is editable in your License Management Org (LMO) and read-only in your customer's org or the other way around.
masterLabel	string	The feature parameter name that appears in the user interface.
value	boolean	The default value for this feature parameter. You can reference this value in your code, just like you reference other values in a subscriber's org.

FeatureParameterDataFlowDirection

Represents the direction of the data flow between your License Management Org (LMO) and the customer's org.

Field Name	Field Type	Description
FeatureParameterDataFlowDirection	string	After a package containing the components is installed, indicates whether the feature parameter's value is editable in your License Management Org (LMO) and read-only in your customer's org or the other way around. Valid values are: • LmoToSubscriber • SubscriberToLmo

Declarative Metadata Sample Definition

The following is an example of a FeatureParameterBoolean component.

```xml
<?xml version="1.0" encoding="UTF-8"?>
<FeatureParameterBoolean xmlns="http://soap.sforce.com/2006/04/metadata">
    <dataflowDirection>SubscriberToLmo</dataflowDirection>
    <masterLabel>Budget Tracking Enabled</masterLabel>
    <value>false</value>
</FeatureParameterBoolean>
```

The following is an example package.xml that references the previous definition (and the definitions for the other feature parameter types).

```xml
<?xml version="1.0" encoding="UTF-8"?>
<Package xmlns="http://soap.sforce.com/2006/04/metadata">
    <types>
        <members>*</members>
        <name>FeatureParameterBoolean</name>
    </types>
    <types>
        <members>*</members>
        <name>FeatureParameterDate</name>
    </types>
    <types>
        <members>*</members>
        <name>FeatureParameterInteger</name>
    </types>
    <version>41.0</version>
</Package>
```

Wildcard Support in the Manifest File

This metadata type supports the wildcard character `*` (asterisk) in the `package.xml` manifest file. For information about using the manifest file, see Deploying and Retrieving Metadata with the Zip File.

FeatureParameterDate

Represents a date feature parameter in a packaging org that has access to the Feature Management App (FMA). Feature parameters let you drive app behavior and track activation metrics in subscriber orgs that install your package. This type extends the Metadata metadata type and inherits its `fullName` field.

File Suffix and Directory Location

FeatureParameterDate components have the suffix `.featureParameterDate`. The components are stored in the `featureParameters` folder, which contains components for all the feature parameter metadata types.

Version

FeatureParameterDate components are available in API version 41.0 and later.

Special Access Rules

Available only in packaging orgs that have access to the Feature Management App (FMA). For details, see Manage Features in the *ISVforce Guide*.

Fields

Field Name	Field Type	Description
`dataFlowDirection`	FeatureParameterDataFlowDirection	After a package containing the components is installed, indicates whether the feature parameter's value is editable in your License Management Org (LMO) and read-only in your customer's org or the other way around.
`masterLabel`	string	The feature parameter name that appears in the user interface.

Field Name	Field Type	Description
`value`	date	The default value for this feature parameter. You can reference this value in your code, just like you reference other values in a subscriber's org.

FeatureParameterDataFlowDirection

Represents the direction of the data flow between your License Management Org (LMO) and the customer's org.

Field Name	Field Type	Description
`FeatureParameterDataFlowDirection`	string	After a package containing the components is installed, indicates whether the feature parameter's value is editable in your License Management Org (LMO) and read-only in your customer's org or the other way around. Valid values are: - `LmoToSubscriber` - `SubscriberToLmo`

Declarative Metadata Sample Definition

The following is an example of a FeatureParameterDate component.

```
<?xml version="1.0" encoding="UTF-8"?>
<FeatureParameterDate xmlns="http://soap.sforce.com/2006/04/metadata">
    <dataflowDirection>SubscriberToLmo</dataflowDirection>
    <masterLabel>Activation Date</masterLabel>
    <value>2017-10-23</value>
</FeatureParameterDate>
```

The following is an example `package.xml` that references the previous definition (and the definitions for the other feature parameter types).

```
<?xml version="1.0" encoding="UTF-8"?>
<Package xmlns="http://soap.sforce.com/2006/04/metadata">
    <types>
        <members>*</members>
        <name>FeatureParameterBoolean</name>
    </types>
    <types>
        <members>*</members>
        <name>FeatureParameterDate</name>
    </types>
    <types>
        <members>*</members>
        <name>FeatureParameterInteger</name>
    </types>
    <version>41.0</version>
</Package>
```

Wildcard Support in the Manifest File

This metadata type supports the wildcard character `*` (asterisk) in the `package.xml` manifest file. For information about using the manifest file, see Deploying and Retrieving Metadata with the Zip File.

FeatureParameterInteger

Represents an integer feature parameter in a packaging org that has access to the Feature Management App (FMA). Feature parameters let you drive app behavior and track activation metrics in subscriber orgs that install your package. This type extends the Metadata metadata type and inherits its `fullName` field.

File Suffix and Directory Location

FeatureParameterInteger components have the suffix `.featureParameterInteger`. The components are stored in the `featureParameters` folder, which contains components for all the feature parameter metadata types.

Version

FeatureParameterInteger components are available in API version 41.0 and later.

Special Access Rules

Available only in packaging orgs that have access to the Feature Management App (FMA). For details, see Manage Features in the *ISVforce Guide*.

Fields

Field Name	Field Type	Description
`dataFlowDirection`	FeatureParameterDataFlowDirection	After a package containing the components is installed, indicates whether the feature parameter's value is editable in your License Management Org (LMO) and read-only in your customer's org or the other way around.
`masterLabel`	string	The feature parameter name that appears in the user interface.
`value`	int	The default value for this feature parameter. You can reference this value in your code, just like you reference other values in a subscriber's org.

FeatureParameterDataFlowDirection

Represents the direction of the data flow between your License Management Org (LMO) and the customer's org.

Field Name	Field Type	Description
`FeatureParameterDataFlowDirection`	string	After a package containing the components is installed, indicates whether the feature parameter's value is editable in your License Management Org (LMO) and read-only in your customer's org or the other way around. Valid values are: • `LmoToSubscriber` • `SubscriberToLmo`

Declarative Metadata Sample Definition

The following is an example of a FeatureParameterInteger component.

```
<?xml version="1.0" encoding="UTF-8"?>
<FeatureParameterInteger xmlns="http://soap.sforce.com/2006/04/metadata">
    <dataflowDirection>SubscriberToLmo</dataflowDirection>
    <masterLabel>Current Project Count</masterLabel>
    <value>42</value>
</FeatureParameterInteger>
```

The following is an example `package.xml` that references the previous definition (and the definitions for the other feature parameter types).

```
<?xml version="1.0" encoding="UTF-8"?>
<Package xmlns="http://soap.sforce.com/2006/04/metadata">
    <types>
        <members>*</members>
        <name>FeatureParameterBoolean</name>
    </types>
    <types>
        <members>*</members>
        <name>FeatureParameterDate</name>
    </types>
    <types>
        <members>*</members>
        <name>FeatureParameterInteger</name>
    </types>
    <version>41.0</version>
</Package>
```

Wildcard Support in the Manifest File

This metadata type supports the wildcard character `*` (asterisk) in the `package.xml` manifest file. For information about using the manifest file, see Deploying and Retrieving Metadata with the Zip File.

FlexiPage

Represents the metadata associated with a Lightning page. A Lightning page represents a customizable screen made up of regions containing Lightning components.

Note: A Lightning page region can contain up to 100 components.

This type extends the Metadata metadata type and inherits its `fullName` field.

Note: These pages are known as FlexiPages in the API, but are referred to as Lightning pages in the rest of the Salesforce documentation and UI.

Lightning pages are used in several places.

- In the Salesforce app, a Lightning page is the home page for an app that appears in the navigation menu.
- In Lightning Experience, Lightning pages can be used:
  - To customize the layout of record pages, the Salesforce Home page, and the Email Application pane in the Outlook and Gmail integrations.
  - As the home page for an app.
  - As the utility bar for a Lightning app.

For more information on Lightning pages, see Salesforce Help.

File Suffix and Directory Location

FlexiPage components have the suffix `.flexipage` and are stored in the `flexipages` folder.

Version

FlexiPage components are available in API version 29.0 and later.

Fields

Field Name	Field Type	Description
`description`	string	The optional description text of the Lightning page.
`flexiPageRegions`	FlexiPageRegion[]	The list of regions of a page.
`masterLabel`	string	Required. The label for the Lightning page, which displays in Setup.
`pageTemplate`	string	Deprecated. Use this field in API versions 33.0 to 38.0 only. In later versions, use `template`. Required. The template associated with the Lightning page.
`parentFlexiPage`	string	The name of the Lightning page that this page inherits behavior from. This field is available in API version 37.0 or later.

Field Name	Field Type	Description
`platformActionList`	PlatformActionList	The list of all actions, and their order, that display on a Lightning app page. In the Salesforce mobile app, the actions appear in the action bar. This field is available in API version 34.0 and later.
`quickActionList`	QuickActionList	The list of quick actions associated with the Lightning page.
`sobjectType`	string	The object the Lightning page is associated with. For Lightning pages of type `AppPage` or `HomePage`, this field is `null`. After the value of this field is set, it can't be changed. This field is available in API version 37.0 or later.
`template`	FlexiPageTemplateInstance	Required. The template associated with the Lightning page. This field is available in API version 39.0 and later.
`type`	FlexiPageType (enumeration of type string)	Required. The type of a page. In API versions 32.0 through 36.0, this field can only have a value of `AppPage`. Valid values are: • `AppPage`—A Lightning page that is used as the home page for a custom app. • `CommAppPage`—A Lightning page that is used to represent a custom page, as created in the Experience Builder, in Communities. This value is available in API version 37.0 and later. • `CommFlowPage` A Lightning page used to override a flow page, as created in the Experience Builder, in Communities. This value is available in API version 45.0 and later. • `CommForgotPasswordPage`—A Lightning page that's used to override a forgot-password page, as created in Experience Builder, in Communities. This value is available in API version 39.0 and later. • `CommFlowPage`—An out-of-the-box flow page, as created in Experience Builder, in Communities. This value is available in API version 45.0 and later. • `CommGlobalSearchResultPage` A Lightning page used to override the global search result page, as created in Experience Builder, in Communities. This value is available in API version 41.0 and later. • `CommLoginPage`—A Lightning page that's used to override the login page, as created in Experience Builder, in Communities. This value is available in API version 39.0 and later.

Field Name	Field Type	Description
		• `CommNoSearchResultsPage`—A Lightning communities page for B2B searches that return no results. The URL for this page is `no-results/:term`. The page starts out empty. You can add any component to it that accepts parameters to achieve the desired "no results" experience. For example, you can place an HTML Editor component or CMS components for recommendations, banners, help and support. This value is available in API version 48.0 and later. • `CommObjectPage`—A Lightning page used to override an object page, as created in Experience Builder, in Communities. This value is available in API version 38.0 and later. • `CommQuickActionCreatePage`—A Lightning page used to override the create record page, as created in Experience Builder, in Communities. This value is available in API version 38.0 and later. • `CommRecordPage`—A Lightning page used to override a record page, as created in the Experience Builder, in Communities. This value is available in API version 38.0 and later. • `CommRelatedListPage`—A Lightning page used to override a related list page, as created in the Experience Builder, in Communities. This value is available in API version 38.0 and later. • `CommSearchResultPage`—A Lightning page used to override the search result page, as created in Experience Builder, in Communities. This value is available in API version 38.0 and later. • `CommSelfRegisterPage`—A Lightning page used to override the self-registration page, as created in Experience Builder, in Communities. This value is available in API version 39.0 and later. • `CommThemeLayoutPage`—A Lightning page used to override a theme layout page, as created in the Experience Builder, in Communities. This value is available in API version 38.0 and later. • `EmbeddedServicePage` This value is available in API version 45.0 and later. • `EmailTemplatePage` — a page that contains the builder markup for your email template. When you edit an email template in the builder, the FlexiPage object remembers where you put the components. This value is reserved for future use.

Field Name	Field Type	Description
		- `HomePage`—A Lightning page that is used to override the Home page in Lightning Experience. This value is available in API version 37.0 and later. - `MailAppAppPage`—An email application pane used to override the default layout in the Outlook and Gmail integrations. This value is available in API version 38.0 and later. - `RecordPage`—A Lightning page used to override an object record page in Lightning Experience. This value is available in API version 37.0 and later. - `RecordPreview` A Lightning page used to override standard lookup previews when hovering over previewable records in Lightning Experience.This value is available in API version 45.0 and later. - `UtilityBar`—A Lightning page used as the utility bar in Lightning Experience apps. This value is available in API version 38.0 and later. This field is available in API version 32.0 and later.

FlexiPageRegion

FlexiPage Region represents the properties of a region of a page. A region can contain a record list component or a recent items component that can be scoped to a set of entities.

Field Name	Field Type	Description
`appendable`	RegionFlagStatus (enumeration of type string)	This field is available in Communities in API 45.0 or later, but is reserved for future use for all other areas. Valid values are: - `disabled` - `enabled` This field is assessed in combination with `replaceable` and `prependable` - If all the properties are set to `enabled`, the region is unlocked - If all the properties are set to `disabled`, the region is locked - If none of the properties are specified OR any of these three properties are missing, the region is unlocked. This field is available in API version 35.0 or later.
`componentInstances`	ComponentInstance[]	Properties and name of the component instance.

Field Name	Field Type	Description
`mode`	FlexiPageRegionMode (enumeration of type string)	This field is reserved for future use. Valid values are: • `Append` • `Prepend` • `Replace` This field is available in API version 35.0 or later.
`name`	string	Required. Unique name of the FlexiPage region.
`prependable`	RegionFlagStatus (enumeration of type string)	This field is available in Communities in API 45.0 or later, but is reserved for future use for all other areas. Valid values are: • `disabled` • `enabled` This field is assessed in combination with `appendable` and `replaceable`. • If all the properties are set to `enabled`, the region is unlocked • If all the properties are set to `disabled`, the region is locked • If none of the properties are specified OR any of these three properties are missing, the region is unlocked. This field is available in API version 35.0 or later.
`replaceable`	RegionFlagStatus (enumeration of type string)	This field is available in Communities in API 45.0 or later, but is reserved for future use for all other areas. Valid values are: • `disabled` • `enabled` This field is assessed in combination with `appendable` and `prependable`. • If all the properties are set to `enabled`, the region is unlocked • If all the properties are set to `disabled`, the region is locked • If none of the properties are specified OR any of these three properties are missing, the region is unlocked. This field is available in API version 35.0 or later.

Field Name	Field Type	Description
`type`	FlexiPageRegionType (enumeration of type string)	Required. The type of FlexiPage region. Valid values are: • `Background`—Represents a region for background utility items, which aren't visible in the UI. Supported for utility bars only. • `Facet` • `Region` This field is available in API version 35.0 or later.

ComponentInstance

Instance of a component in a page, such as a filter list.

Field Name	Field Type	Description
`componentInstanceProperties`	ComponentInstanceProperty[]	The value of a single property in a component instance.
`componentName`	string	Required. The name of a single instance of a component.
`visibilityRule`	UiFormulaRule	A set of one or more filters that define the conditions under which the component displays on the page. If the rule evaluates to `true`, the component displays on the page. If `false`, it doesn't display. If this field is `null`, the component displays by default. This field is available in API version 41.0 and later.

ComponentInstanceProperty

Value of a single property in a component instance.

Field Name	Field Type	Description
`name`	string	Name of the property, unique within the component instance. For Lightning components, this value is the `<aura:attribute>` as defined in the `.cmp` file.
`type`	ComponentInstancePropertyTypeEnum (enumeration of type string)	If this field value is `null`, then the ComponentInstanceProperty values apply to the Lightning component. If this field value is `decorator`, then the ComponentInstanceProperty values apply to the *component decorator* for the Lightning component. The component decorator is a wrapper around a Lightning component. The decorator can apply more capabilities to the

Field Name	Field Type	Description
		component when it renders on a specific page in Lightning Experience. For example, you can configure a component decorator around a component on the Lightning Experience utility bar to set the component's height or width when opened. The UtilityBar is the only page type that supports component decorators. Valid values are: • decorator This field is available in API version 38.0 or later.
value	string	Reference or value of the property. When defining a Related List component, to use a parent record set the parentFieldApiName value to ***object.field_name***. If you don't want to use a parent record, set the value to ***object.***Id.

When you give a standard label to a tab in a Tabs component—such as Activity, Collaborate, or Details—and when the name field is set to title, the value field uses a system-defined value instead of the label. Here are some examples of the system-defined values:

- Standard.Tab.activity
- Standard.Tab.collaborate
- Standard.Tab.detail
- Standard.Tab.feed
- Standard.Tab.preview
- Standard.Tab.relatedLists

For example, let's say you have a Lightning page that contains a tab with the standard label "Activity". If you query the definition that page, you see the system-defined name of the tab, not the label, in value.

```
<componentInstances>
  <componentInstanceProperties>
    <name>title</name>
    <value>Standard.Tab.activity</value>
  </componentInstanceProperties>
    <componentName>flexipage:tab</componentName>
</componentInstances>
```

UiFormulaRule

A set of one or more filters that define the conditions under which a component displays on a Lightning page. For example, you could construct a filter that causes a rich text component on an opportunity page to display only when the Amount is greater than $1,000,000. Available in API version 41.0 and later.

Field Name	Field Type	Description
`booleanFilter`	string	Specifies advanced filter conditions such as `1 AND 2`.
`criteria`	UiFormulaCriterion[]	List of one or more filters that, when evaluated, determine component visibility.

FlexiPageTemplateInstance

FlexiPageTemplateInstance represents an instance of a Lightning page template.

Field Name	Field Type	Description
`name`	string	Required. The name of a single instance of a template.
`properties`	ComponentInstanceProperty[]	The value of a single property in a template instance. Valid only for `CommThemeLayoutPage`. Contains a name and value pair for each theme layout property associated with the page template. In Experience Builder, the theme layout and its properties appear in the Theme area.

PlatformActionList

PlatformActionList represents the list of actions, and their order, that display on a Lightning app page. Available in API version 34.0 and later.

Field Name	Field Type	Description
`actionListContext`	PlatformActionListContext (enumeration of type string)	Required. The context of the action list. Valid values are: • `ActionDefinition` • `Assistant` • `BannerPhoto` • `Chatter` • `Dockable` • `FeedElement` • `Flexipage` • `Global` • `ListView` • `ListViewDefinition` • `ListViewRecord` • `Lookup` • `MruList` • `MruRow`

Field Name	Field Type	Description
		- `ObjectHomeChart` - `Photo` - `Record` - `RecordEdit` - `RelatedList` - `RelatedListRecord`
`platformActionListItems`	PlatformActionListItem[]	The actions in the PlatformActionList.
`relatedSourceEntity`	string	When the `ActionListContext` is RelatedList or RelatedListRecord, this field represents the API name of the related list to which the action belongs.

PlatformActionListItem

PlatformActionListItem represents an action in the PlatformActionList. Available in API version 34.0 and later.

Field Name	Field Type	Description
`actionName`	string	Required. The API name for the action in the list.
`actionType`	PlatformActionType (enumeration of type string)	Required. The type of action. Valid values are: - `ActionLink`—An indicator on a feed element that targets an API, a web page, or a file, represented by a button in the Salesforce Chatter feed UI. - `CustomButton`—When clicked, opens a URL or a Visualforce page in a window or executes JavaScript. - `InvocableAction` - `ProductivityAction`—Productivity actions are predefined and attached to a limited set of objects. Productivity actions include Send Email, Call, Map, View Website, and Read News. Except for the Call action, you can't edit or delete productivity actions. - `QuickAction`—A global or object-specific action. - `StandardButton`—A predefined Salesforce button such as New, Edit, and Delete.
`sortOrder`	int	Required. The placement of the action in the list.
`subtype`	string	The subtype of the action. For quick actions, the subtype is `QuickActionType`. For custom buttons, the subtype is `WebLinkTypeEnum`. For action links, subtypes are `Api`, `ApiAsync`, `Download`, and `Ui`. Standard buttons and productivity actions have no subtype.

UiFormulaCriterion

A single filter that when evaluated, helps define component visibility on a Lightning page. Available in API version 41.0 and later.

Field Name	Field Type	Description
`leftValue`	string	Required. The field upon which the filter is based. For example, `AMOUNT`.
`operator`	string	Required. Defines the operator used to filter the data. Valid values are: • `CONTAINS` • `EQUAL` • `NE`—not equal • `GT`—greater than • `GE`—greater than or equal • `LE`—less than or equal • `LT`—less than
`rightValue`	string	The value by which you want to evaluate the component's visibility. For example, `1000000`.

You can use these expressions in the `leftValue` field when setting filters for component visibility.

- `{!$Client.FormFactor}`—Use this expression to control component visibility based on the device the page is being rendered on. Valid values are `Small` (phone), `Medium` (tablet), and `Large` (Lightning Experience desktop). Setting the value to `Small` for record pages is supported only in orgs that are enabled for the new Salesforce mobile app. This expression is supported for app pages in API version 41.0 and later, and record pages in API version 47.0 and later.
- `{!$Permission.CustomPermission.`***`permissionName`***`}`—Use this expression to control component visibility based on the custom permissions of the user viewing the Lightning page. Supported for app, Home, and record pages only.
- `{!$Permission.StandardPermission.`***`permissionName`***`}`—Use this expression to control component visibility based on the standard permissions of the user viewing the Lightning page. Supported for app, Home, and record pages only.
- `{!Record.`***`field`***`}`—Supported for record pages only.
- `{!$User.`***`field`***`}`—Supported for app, Home, and record pages only.

For example, to display a component only when it renders on a phone, add this filter: `{!$Client.FormFactor} EQUAL "SMALL"`. Or, to display a component only to the System Administrator, use `{!$User.Profile.Name} EQUAL "System Administrator"`.

Expressions in component visibility rules can span no more than five fields. For example, `{!Record.Account.Owner.Manager.Manager.Manager.LastName}` has six spans and therefore isn't supported.

Declarative Metadata Sample Definition

Here's a sample XML FlexiPage component definition for a custom opportunity record page. It includes a tab set and a rich text component with visibility rules assigned to it.

Note: As a community page, three initial regions in the definition show the `header` region as locked, the `content` region as unlocked, and the `footer` region as unlocked.

```
<?xml version="1.0" encoding="UTF-8"?>
<FlexiPage xmlns="http://soap.sforce.com/2006/04/metadata">
        <componentInstances>
            <componentInstanceProperties>
                <name>collapsed</name>
                <value>false</value>
            </componentInstanceProperties>
            <componentInstanceProperties>
                <name>numVisibleActions</name>
                <value>3</value>
            </componentInstanceProperties>
            <componentName>force:highlightsPanel</componentName>
        </componentInstances>
    <flexiPageRegions>
        <appendable>disabled</appendable>
        <name>header</name>
        <prependable>disabled</prependable>
        <replaceable>disabled</replaceable>
        <type>Region</type>
    </flexiPageRegions>
    <flexiPageRegions>
        <appendable>enabled</appendable>
        <componentInstances>
            <componentName>forceCommunity:fileDetailPage</componentName>
        </componentInstances>
        <name>content</name>
        <prependable>enabled</prependable>
        <replaceable>enabled</replaceable>
        <type>Region</type>
    </flexiPageRegions>
    <flexiPageRegions>
                <name>footer</name>
        <type>Region</type>
    </flexiPageRegions>

    <flexiPageRegions>
        <componentInstances>
            <componentName>force:detailPanel</componentName>
        </componentInstances>
        <name>facet-82dce3a7-1222-4948-8b41-533cdd60aaa4</name>
        <type>Facet</type>
    </flexiPageRegions>
    <flexiPageRegions>
        <componentInstances>
            <componentName>runtime_sales_activities:activityPanel</componentName>
        </componentInstances>
        <name>facet-4bdf038a-4568-4c4e-b220-b5ec47f656e9</name>
        <type>Facet</type>
    </flexiPageRegions>
    <flexiPageRegions>
        <componentInstances>
```

```
            <componentName>wits:newsPanel</componentName>
        </componentInstances>
        <name>facet-8d5ecfff-4590-49c0-a09e-3df84179f642</name>
        <type>Facet</type>
    </flexiPageRegions>
    <flexiPageRegions>
        <componentInstances>
            <componentName>force:relatedListContainer</componentName>
        </componentInstances>
        <name>facet-551d8770-5093-4222-927e-9138061ebbbf</name>
        <type>Facet</type>
    </flexiPageRegions>
    <flexiPageRegions>
        <componentInstances>
            <componentInstanceProperties>
                <name>body</name>
                <value>facet-82dce3a7-1222-4948-8b41-533cdd60aaa4</value>
            </componentInstanceProperties>
            <componentInstanceProperties>
                <name>title</name>
                <value>Standard.Tab.detail</value>
            </componentInstanceProperties>
            <componentName>flexipage:tab</componentName>
        </componentInstances>
        <componentInstances>
            <componentInstanceProperties>
                <name>body</name>
                <value>facet-4bdf038a-4568-4c4e-b220-b5ec47f656e9</value>
            </componentInstanceProperties>
            <componentInstanceProperties>
                <name>title</name>
                <value>Standard.Tab.activity</value>
            </componentInstanceProperties>
            <componentName>flexipage:tab</componentName>
        </componentInstances>
        <componentInstances>
            <componentInstanceProperties>
                <name>body</name>
                <value>facet-8d5ecfff-4590-49c0-a09e-3df84179f642</value>
            </componentInstanceProperties>
            <componentInstanceProperties>
                <name>title</name>
                <value>Recent Items</value>
            </componentInstanceProperties>
            <componentName>flexipage:tab</componentName>
        </componentInstances>
        <componentInstances>
            <componentInstanceProperties>
                <name>active</name>
                <value>true</value>
            </componentInstanceProperties>
            <componentInstanceProperties>
                <name>body</name>
                <value>facet-551d8770-5093-4222-927e-9138061ebbbf</value>
            </componentInstanceProperties>
            <componentInstanceProperties>
                <name>title</name>
                <value>Standard.Tab.relatedLists</value>
            </componentInstanceProperties>
            <componentName>flexipage:tab</componentName>
        </componentInstances>
        <name>facet-f433d4c6-94e6-4de4-90fc-83501abf3536</name>
        <type>Facet</type>
    </flexiPageRegions>
    <flexiPageRegions>
        <componentInstances>
            <componentInstanceProperties>
                <name>tabs</name>
                <value>facet-f433d4c6-94e6-4de4-90fc-83501abf3536</value>
            </componentInstanceProperties>
            <componentName>flexipage:tabset</componentName>
        </componentInstances>
        <name>main</name>
        <type>Region</type>
    </flexiPageRegions>
    <flexiPageRegions>
        <componentInstances>
            <componentInstanceProperties>
                <name>hideUpdateButton</name>
                <value>false</value>
            </componentInstanceProperties>
            <componentName>runtime_sales_pathassistant:pathAssistant</componentName>
        </componentInstances>
        <name>subheader</name>
        <type>Region</type>
    </flexiPageRegions>
    <flexiPageRegions>
        <componentInstances>
            <componentInstanceProperties>
                <name>decorate</name>
                <value>true</value>
            </componentInstanceProperties>
            <componentInstanceProperties>
                <name>richTextValue</name>
                <value>&lt;div style=&quot;text-align: center;&quot;&gt;&lt;
                span style=&quot;font-family:georgia,serif;&quot;&gt;&lt;
                span style=&quot;font-size: 18px;&quot;&gt;&lt;b&gt;
                A million dollar opportunity closed! Oh yeah!&lt;/b&gt;&lt;
                /span&gt;&lt;/span&gt;&lt;/div&gt;</value>
            </componentInstanceProperties>
            <componentName>flexipage:richText</componentName>
            <visibilityRule>
                <booleanFilter>1 AND 2</booleanFilter>
                <criteria>
                    <leftValue>{!Record.Amount}</leftValue>
                    <operator>GE</operator>
                    <rightValue>1000000</rightValue>
                </criteria>
                <criteria>
                    <leftValue>{!Record.StageName}</leftValue>
                    <operator>EQUAL</operator>
                    <rightValue>Closed Won</rightValue>
                </criteria>
            </visibilityRule>
        </componentInstances>
        <componentInstances>
            <componentName>forceChatter:recordFeedContainer</componentName>
        </componentInstances>
        <name>sidebar</name>
        <type>Region</type>
    </flexiPageRegions>
    <masterLabel>New Opportunity Page</masterLabel>
    <sobjectType>Opportunity</sobjectType>
    <template>
        <name>flexipage:recordHomeWithSubheaderTemplateDesktop</name>
    </template>
    <type>RecordPage</type>
</FlexiPage>
```

And, here’s the sample `package.xml` file that references the FlexiPage component definition:

```
<?xml version="1.0" encoding="UTF-8"?>
<Package xmlns="http://soap.sforce.com/2006/04/metadata">
    <fullName>New Opportunity Page Package</fullName>
    <description>Contains an opportunity page with a rich text component that has rules
assigned</description>
    <types>
        <members>New_Opportunity_Page</members>
        <name>FlexiPage</name>
    </types>
    <version>41.0</version>
</Package>
```

Wildcard Support in the Manifest File

This metadata type supports the wildcard character `*` (asterisk) in the `package.xml` manifest file. For information about using the manifest file, see Deploying and Retrieving Metadata with the Zip File.

Flow

Represents the metadata associated with a flow. With Flow, you can create an application that navigates users through a series of screens to query and update records in the database. You can also execute logic and provide branching capability based on user input to build dynamic applications.

For information about the corresponding UI-based flow building tool, see “Flow Builder” in Salesforce Help.

When using Metadata API to work with flows, consider that:

- You can’t use Metadata API to access a flow installed from a managed package, unless the flow is a template.
- Spaces in flow file names can cause errors at deployment. Heading and trailing spaces are allowed, but are trimmed during deployment.
- When using Metadata API, you can deploy changes to active flows in specific situations.

  - The org isn't a production org (for example, a scratch org or sandbox). The active flow is the latest version.
  - The org is a production org that enabled the Deploy processes and flows as active preference. The active flow is the latest version.

  After you deploy changes to an active flow, the flow's detail page displays a new flow version that's active. The new version includes your changes. For example, version 3 of myflow is active and the latest version. After you change and deploy version 3, the myflow's detail page displays version 4 as active.

- You can delete a flow version as long as it isn't active and has no paused interviews. If the flow version has paused interviews, wait for those interviews to resume and finish, or delete them.

Warning: Don't edit the metadata of retrieved Process Builder processes (Flow components whose processType is Workflow or InvocableProcess.) If you deploy process metadata that you've edited, you might not be able to open the process in the target org.

Declarative Metadata File Suffix and Directory Location

Flows are stored in the `Flow` directory of the corresponding package directory. The file name matches the flow's unique full name, and the extension is `.flow`.

Version

The flow Metadata API is available in API version 24.0 and later.

Flow

This metadata type represents a valid definition of a flow. This type extends the Metadata metadata type and inherits its `fullName` field.

Field Name	Field Type	Description
actionCalls	FlowActionCall[]	An array of nodes that define calls to actions. This field is available in API version 31.0 and later.
apexPluginCalls	FlowApexPluginCall[]	An array of nodes that define calls to Apex plug-ins.
assignments	FlowAssignment[]	An array of assignment nodes.
choices	FlowChoice[]	An array of static choice options.
constants	FlowConstant[]	An array of constants.
decisions	FlowDecision[]	An array of decision nodes.
description	string	Description of the flow.
dynamicChoiceSets	FlowDynamicChoiceSet[]	An array that constructs a set of choice options based on a database lookup.
formulas	FlowFormula[]	An array of formulas.

Field Name	Field Type	Description
fullName	string	Required; inherited from the Metadata component. Name of the file in Metadata API. A unique name for the flow that contains only underscores and alphanumeric characters. The name must be unique across the org, begin with a letter, not include spaces, not end with an underscore, and not contain two consecutive underscores. To deploy or retrieve a version, you can specify the version number. For example, `sampleFlow-3` specifies version 3 of the flow whose unique name is sampleFlow. If you don't specify a version number, the flow is the latest version. In API version 43.0 and earlier, this field included the version number. In API version 44 and later, this field no longer includes the version number.
interviewLabel	string	Label for the interview. This label helps users and administrators differentiate interviews from the same flow. In the user interface, this label appears in the Paused Flow Interviews component on the user's Home tab and in the list of paused flow interviews in Setup.
isAdditionalPermissionRequiredToRun	boolean	Indicates whether to override the default behavior and restrict access to enabled profiles or permission sets (`true`) or not (`false`). The default value is (`false`). Available in API version 47.0 and later.
isTemplate	boolean	Indicates whether the process or flow is a template. When installed from managed packages, processes and flows can't be viewed or cloned by subscribers because of intellectual property (IP) protection. But when those processes and flows are templates, subscribers can open them in a builder, clone them, and customize the clones. Available in API version 45.0 and later. Default: false
label	string	Required. Label for the flow.
loops	FlowLoop[]	An array of nodes for iterating through collections. This field is available in API version 30.0 and later.
processMetadataValues	FlowMetadataValue[]	Metadata values for the flow. This field is available in API version 31.0 and later.
processType	FlowProcessType (enumeration of type string)	The type of the flow, as determined by the active version (or latest version, if there's no active version). Valid values are: • `Appointments`—A flow for Lightning Scheduler. This value is available in API version 44.0 and later.

Field Name	Field Type	Description
		• `AutoLaunchedFlow`—A flow that doesn't require user interaction. • `ContactRequestFlow`—A flow that lets customers request that customer support get back to them. This flow is used to create contact request records. This value is available in API version 45.0 and later. • `CustomEvent`—A process that is invoked when it receives a platform event message. In the UI, it's an event process. This value is available in API version 41.0 and later. • `FieldServiceMobile`—A flow for the Field Service Lightning mobile app. This value is available in API version 39.0 and later. • `FieldServiceWeb`—A flow for embedded Appointment Booking. Its UI label is Field Service Embedded Flow. This value is available in API version 41.0 and later. • `Flow`—A flow that requires user interaction because it contains one or more screens or local actions, choices, or dynamic choices. In the UI and Salesforce Help, it's a screen flow. Screen flows can be launched from the UI, such as with a flow action, Lightning page, or web tab. • `InvocableProcess`—A process that can be invoked by another process or the Invocable Actions resource in REST API. This value is available in API version 38.0 and later. • `Survey`—A flow for Salesforce Surveys. From the UI, this type of flow is created in Survey Builder. This value is available in API version 42.0 and later. • `Workflow`—A process that is invoked when a record is created or edited. In the UI and Salesforce Help, it's a record change process. Across flow versions, you can change the type only from `Flow` to `AutolaunchedFlow` or vice versa. Before you change the flow type, make sure that the flow contains only elements, resources, and functionality that the new flow type supports. These values are reserved for future or Salesforce internal use. • `ActionCadenceFlow` • `ActionPlan` • `CartAsyncFlow` • `CheckoutFlow` • `DecisionStudio` • `DigitalForm` • `Journey` • `JourneyBuilderIntegration`

Field Name	Field Type	Description
		- `LoginFlow` - `ManagedContentFlow` - `OrchestrationFlow` - `SalesEntryExperienceFlow` - `SurveyEnrich` - `TransactionSecurityFlow` - `UserProvisioningFlow` This field is available in API version 31.0 and later.
`recordCreates`	FlowRecordCreate[]	An array of nodes for creating records in the database.
`recordDeletes`	FlowRecordDelete[]	An array of nodes for deleting records in the database.
`recordLookups`	FlowRecordLookup[]	An array of nodes for looking up records in the database.
`recordUpdates`	FlowRecordUpdate[]	An array of nodes for updating records in the database.
`runInMode`	FlowRunInMode (enumeration of type string)	The context that the flow runs in. Valid values are: - `DefaultMode`—How the flow is launched determines whether the flow runs in user context or in system context. In the UI, this value appears as **User or System Context—Depends on How Flow is Launched**. - `SystemModeWithSharing`—The flow respects org-wide default settings, role hierarchies, sharing rules, manual sharing, teams, and territories. But the flow doesn't respect object permissions, field-level access, or other permissions of the running user. In the UI, this value appears as **System Context with Sharing—Enforces Record-Level Access**. This field is available in API version 48.0 and later.
`screens`	FlowScreen[]	An array of screen nodes.
`stages`	FlowStage[]	An array of stages that you can use throughout the flow. This field is available in API version 42.0 and later.
`start`	FlowStart	Represents the flow's Start element, which specifies how and when the flow starts. This field is available in API version 47.0 and later.
`startElementReference`	string	Specifies which node or element is the starting point in the flow. This field isn't used in flows that are created or saved in Flow Builder in Winter '20 and later. Those flows use the `start` field instead to specify how the flow starts.
`status`	FlowVersionStatus (enumeration of type string)	The activation status of the flow. Valid values are: - `Active`

Field Name	Field Type	Description
		- Draft—In the UI, this status appears as Inactive. - Obsolete—In the UI, this status appears as Inactive. - InvalidDraft—In the UI, this status appears as Draft.
steps	FlowStep[]	An array of step nodes.
subflows	FlowSubflow[]	An array of subflows. This field is available in API version 25.0 and later.
textTemplates	FlowTextTemplate[]	An array of text templates.
variables	FlowVariable[]	An array of variable definitions.
waits	FlowWait[]	An array of wait nodes. This field is available in API version 32.0 and later.

FlowActionCall

Defines a call to an action from the flow. It extends FlowNode.

Available in API version 31.0 and later.

Field Name	Field Type	Description
actionName	string	Required. Name for the action. Must be unique across actions with the same actionType.
actionType	InvocableActionType (enumeration of type string)	Required. The action type. Valid values are: - activateSessionPermSet—activates a session-based permission set for the running user. - addMessageToQuipChat—adds a Quip message to an existing chat room. Available in API version 46.0 and later. - addMessageToQuipDocument—adds a Quip message to an existing Quip document, spreadsheet, or slide . Available in API version 46.0 and later. - addQuipDocumentToFolder—adds an existing Quip document, spreadsheet, or slide to an existing folder. Available in API version 46.0 and later. - addUsersToQuipDocument—adds users, identified by their email addresses, to an existing Quip document, spreadsheet, or slide. Available in API version 46.0 and later. - addUsersToQuipChat—adds users, identified by their email addresses, to an existing Quip chat room. Available in API version 46.0 and later.

Field Name	Field Type	Description
		• `attachQuipDocumentToRecord`—attaches a Quip document, spreadsheet, or slide to a Salesforce record. Available in API version 46.0 and later. • `apex`—invokes an Apex method that has the @invocableMethod annotation • `archiveKnowledgeArticles`—archives a list of published Knowledge articles. Available in API version 45.0 and later. • `assignKnowledgeArticles`—mass assigns Knowledge articles from article list views. Available in API version 44.0 and later. • `chatterPost`—posts to Chatter • `choosePricebook`—selects a price book • `contactRequestAction`—creates a contact request record. Available in API version 45.0 and later. • `component`—invokes the Lightning component that implements the `lightning:availableForFlowActions` interface and that is referenced by `actionName`. Available in API version 43.0 and later. • `contentWorkspaceEnableFolders`—enables folders in a library • `copyQuipDocument`—creates a copy of an existing Quip document, spreadsheet, or slide and gives it a new title. Available in API version 46.0 and later. • `createDraftFromOnlineKnowledgeArticle`—creates a draft from a published Knowledge article. Available in API version 45.0 and later. • `createQuipChat`—creates a Quip chat room. Available in API version 46.0 and later. • `createQuipDocument`—creates a Quip document, spreadsheet, or slide. Available in API version 46.0 and later. • `createQuipFolder`—creates a Quip folder. Available in API version 46.0 and later. • `customNotificationAction`—sends a custom notification. Available in API version 46.0 and later. • `deactivateSessionPermSet`—deactivates a session-based permission set for the running user. • `deleteKnowledgeArticle`—deletes a draft version (translation or master-language) or an entire archived Knowledge article. Available in API version 46.0 and later.

Field Name	Field Type	Description
		• `editQuipDocument`—modifies the contents of an existing Quip document, spreadsheet, or slide. Available in API version 46.0 and later. • `emailAlert`—sends an email by referencing a workflow email alert • `emailSimple`—sends an email by using flow resources • `externalService`—invokes an External Service operation that makes an HTTP request to an external system made available by an External Service schema registered through Setup. Available in API version 46.0 and later. • `flow`—invokes an autolaunched flow. This action type isn't available for flows with a processType of "Flow" or "AutolaunchedFlow". To invoke an autolaunched flow from one of those types, use FlowSubflow. Available in API version 32.0 and later. • `quickAction`—invokes a QuickAction • `publishKnowledgeArticles`—mass publishes Knowledge articles from article list views. Available in API version 44.0 and later. • `restoreKnowledgeArticleVersion`—restores an archived version of a Knowledge article. Available in API version 45.0 and later. • `submitKnowledgeArticleForTranslation`—submits a published or draft Knowledge article for translation. Available in API version 46.0 and later. • `submit`—submits a record for approval These values are used in Order Management. They are available in API version 48.0 and later. • `cancelFulfillmentOrderItem`—removes items from a fulfillment order. • `cancelOrderItemSummariesPreview`—previews the expected results of canceling order item summaries from an order summary without actually canceling them. • `cancelOrderItemSummariesSubmit`—cancels order item summaries from an order summary. • `createCreditMemoOrderSummary`—creates a credit memo for an order summary. • `createFulfillmentOrder`—creates a fulfillment order for specified order item summaries belonging to an order summary.

Field Name	Field Type	Description
		- `createInvoiceFromFulfillmentOrder`—creates an invoice for a fulfillment order. - `createOrderPaymentSummary`—creates an order payment summary for an authorization or payments belonging to an order summary. - `createOrderSummary`—creates an order summary for an order. - `ensureFundsOrderSummaryAsync`—triggers an asynchronous background process to ensure funds through a payment provider for an invoice belonging to an order summary. - `ensureRefundsOrderSummaryAsync`—triggers an asynchronous background process to ensure refunds through a payment provider for an invoice belonging to an order summary. - `returnOrderItemSummariesPreview`—previews the expected results of returning order item summaries from an order summary without actually returning them. - `returnOrderItemSummariesSubmit`—returns order item summaries from an order summary. For values used in Manufacturing Cloud, seeFlow for Manufacturing Cloud . These values are reserved for future use. - `thanks` - `metricRefresh` - `exportSurveyResponses` - `sendSurveyInvitation`
`connector`	FlowConnector	Specifies which node to execute after this action call.
`dataTypeMappings`	FlowDataTypeMapping[]	An array of data type mappings for input and output values that have the generic sObject data type. This field is available in API version 48.0 and later.
`faultConnector`	FlowConnector	Specifies which node to execute if the action call results in an error.
`inputParameters`	FlowActionCallInputParameter[]	An array of input parameters from the flow to the action.
`outputParameters`	FlowActionCallOutputParameter[]	An array of output parameters from the action to the flow.
`storeOutputAutomatically`	boolean	Indicates whether the action's output parameters are automatically available in the flow without creating any variables. When the value is `true`, you can reference an output parameter by specifying the API name of the Action element in the flow. The default value is `false`. When the

Field Name	Field Type	Description
		value is `false`, create variables manually to store output values from the action. This field is available in API version 48.0 and later.

FlowActionCallInputParameter

Defines an input parameter from the flow to the action. It extends FlowBaseElement and inherits all its fields. Available in API version 31.0 and later.

Field Name	Field Type	Description
`name`	string	Required. Unique name for the input parameter.
`value`	FlowElementReferenceOrValue	Defines the value of the input parameter.

FlowActionCallOutputParameter

Defines an output parameter from the action to the flow. It extends FlowBaseElement and inherits all its fields. Available in API version 31.0 and later.

Field Name	Field Type	Description
`assignToReference`	string	Required. Specifies the variable to which you want to assign the output parameter value.
`name`	string	Required. Unique name for the output parameter.

FlowApexPluginCall

Defines a call to an Apex plug-in from the flow. It extends FlowNode and inherits all its fields.

Field Name	Field Type	Description
`apexClass`	string	Required. The name of the Apex class.
`connector`	FlowConnector	Specifies which node to execute after this Apex plug-in call.
`faultConnector`	FlowConnector	Specifies which node to execute if the Apex plug-in call results in an error.
`inputParameters`	FlowApexPluginCallInputParameter[]	An array of input parameters from the flow to the Apex plug-in.
`outputParameters`	FlowApexPluginCallOutputParameter[]	An array of output parameters from the Apex plug-in to the flow.

FlowApexPluginCallInputParameter

Defines an input parameter from the flow to the Apex plug-in. It extends FlowBaseElement and inherits all its fields.

Field Name	Field Type	Description
name	string	Required. Unique name for the input parameter.
value	FlowElementReferenceOrValue	Defines the value of the input parameter.

FlowApexPluginCallOutputParameter

Defines an output parameter from the Apex plug-in to the flow. It extends FlowBaseElement and inherits all its fields.

Field Name	Field Type	Description
assignToReference	string	Required. Specifies the variable to which you want to assign the output parameter value.
name	string	Required. Unique name for the output parameter.

FlowAssignment

Defines an assignment node that can dynamically change the value of a variable in the flow. It extends FlowNode and inherits all of its fields.

Field Name	Field Type	Description
assignmentItems	FlowAssignmentItem[]	An array of assignment operations that is executed in the given order, starting from the index 0.
connector	FlowConnector	Specifies which node to execute after this assignment node.

FlowAssignmentItem

Defines an operation to apply to a variable. It extends FlowBaseElement and inherits all its fields.

Field Name	Field Type	Description
assignToReference	string	Required. Reference to the variable to which you want to apply the specified operator.
operator	FlowAssignmentOperator (enumeration of type string)	Required. Operation to apply to the variable reference in the assignToReference field. For valid values, see FlowAssignmentOperator.
value	FlowElementReferenceOrValue	Defines the value that you want the operator to apply to the variable reference in the assignToReference field.

FlowAssignmentOperator

An enumeration of type string that specifies the operation to apply to the variable in the `assignToReference` field. See "Flow Operators in Assignment Elements" in Salesforce Help.

Valid values are:

Enumeration Value	Description
`Add`	When the `assignToReference` field is a variable of type number or currency, this operator adds the `value` to the variable. When the `assignToReference` field is a variable of type date, this operator adds the `value` in days to the variable. When the `assignToReference` field is a variable of type string, this operator appends the `value` to the end of the string. When the `assignToReference` field is a variable of type picklist, this operator appends the `value` to the end of the last item in the picklist. When the `assignToReference` field is a variable of type multipicklist, this operator appends the `value` to the end of the last item in the multi-select picklist. To instead add an item to the end of the multi-select picklist, use the `AddItem` operator. When the `assignToReference` field is the `$Flow.ActiveStages` global variable, this operator appends the `value` as a new item at the end of `$Flow.ActiveStages`. When the `assignToReference` field is a collection variable, this operator appends the `value` to the end of the collection. Support for a collection variable as the `value` is available in API version 43.0 and later but only via Metadata API. From Flow Builder, you can't save an Assignment element that contains a collection variable in the Value column for the `Add` operator. The `Add` operator is not supported when the `assignToReference` field is a variable of type boolean, dateTime, or sObject.
`AddAtStart`	Supported only when the `assignToReference` field is a collection variable or the `$Flow.ActiveStages` global variable. Adds the `value` as a new item at the beginning of the collection. When the `value` is a collection variable, the operator adds all items at the beginning of the collection. This operator is available in API version 43.0 and later.
`AddItem`	Supported only when the `assignToReference` field is a variable of type multipicklist. Adds the `value` to the picklist, including the semi-colon that's required to mark a `value` as a separate item. This operator is available in API version 34.0 and later.
`Assign`	Assigns the `value` to the variable in the `assignToReference` field.
`AssignCount`	Supported only when the `value` is a collection variable or the `$Flow.ActiveStages` global variable. Counts the number of stages or items in the collection and assigns that number to the variable in the `assignToReference` field. Corresponds to `equals count` in the user interface. This operator is available in API version 43.0 and later.
`RemoveAfterFirst`	Supported only when the `assignToReference` field is a collection variable or the `$Flow.ActiveStages` global variable. Finds the first instance of the `value` within the

Enumeration Value	Description
	variable in the `assignToReference` field. Removes everything after that first instance from the variable. This operator is available in API version 43.0 and later.
`RemoveAll`	Supported only when the `assignToReference` field is a collection variable or the `$Flow.ActiveStages` global variable. Removes all instances of the `value` from the variable in the `assignToReference` field. When the `value` is a collection variable, the operator removes all instances of each item from the variable in the `assignToReference` field. This operator is available in API version 43.0 and later.
`RemoveBeforeFirst`	Supported only when the `assignToReference` field is a collection variable or the `$Flow.ActiveStages` global variable. Finds the first instance of the `value` within the variable in the `assignToReference` field. Removes everything before that first instance from the variable. This operator is available in API version 43.0 and later.
`RemoveFirst`	Supported only when the `assignToReference` field is a collection variable or the `$Flow.ActiveStages` global variable. Removes the first instance of the `value` from the variable in the `assignToReference` field. This operator is available in API version 43.0 and later.
`RemovePosition`	Supported only when the `assignToReference` field is a collection variable or the `$Flow.ActiveStages` global variable. Removes the item at the specified position. For example, if the collection contains three items (Red, Green, Blue) and the `value` is 2, the second item (Green) is removed from the collection variable. This operator is available in API version 43.0 and later. Make sure that the `value` at run time is a positive integer within the range of the number of items in the collection variable.
`RemoveUncommon`	Supported only when `assignToReference` and `value` are both collection variables. Keeps items that are in both collections and removes the rest from the collection variable in the `assignToReference` field. This operator is available in API version 43.0 and later.
`Subtract`	Supported only when the `assignToReference` field is a variable of type currency, date, or number. When the `assignToReference` field is a variable of type number or currency, this operator subtracts the `value` from the variable. When the `assignToReference` field is a variable of type date, this operator subtracts the `value` in days from the variable.

FlowChoice

A choice resource is a standalone choice option that you can reference or reuse throughout the flow. It extends FlowElement and inherits all of its fields.

Field Name	Field Type	Description
`choiceText`	string	Required. Choice label to display in the screen.

Field Name	Field Type	Description
`dataType`	FlowDataType (enumeration of type string)	Required. Valid types are: • `Currency` • `Date` • `Number` • `String` • `Boolean`
`userInput`	FlowChoiceUserInput	Enables the choice to allow user input when the choice is selected. Not supported for choices in multi-select fields.
`value`	FlowElementReferenceOrValue	Actual value that's used during flow execution, for example, in assignments, calls to Apex plug-ins, and record elements. If null, this choice always has the value of null.

FlowChoiceUserInput

Allows the choice to include a user input field that appears when the choice is selected by the user. User input isn't supported for choices in multi-select fields. It extends FlowBaseElement and inherits all its fields.

Field Name	Field Type	Description
`isRequired`	boolean	Indicates whether users are required to enter something into the field when they select the choice.
`promptText`	string	Text that is displayed to prompt the user for input at runtime. Supports merge fields.
`validationRule`	FlowInputValidationRule	Rule used at runtime to validate the user input.

FlowCondition

Defines a condition for a rule. It extends FlowBaseElement and inherits all its fields.

Field Name	Field Type	Description
`leftValueReference`	string	Required. Unique name of the element that serves as the left side of the condition expression.
`operator`	FlowComparisonOperator (enumeration of type string)	Required. Valid values are: • `EqualTo` • `NotEqualTo` • `GreaterThan` • `LessThan` • `GreaterThanOrEqualTo` • `LessThanOrEqualTo`

Field Name	Field Type	Description
		• `StartsWith` • `EndsWith` • `Contains` • `IsNull` • `WasSet`—This value is available in API version 30.0 and later. • `WasSelected`—Requires a choice on the left side. • `WasVisited`—Requires a node on the left side.
`rightValue`	FlowElementReferenceOrValue	Unique name of an element or the actual value (such as text or a number) for the right side of the condition expression.

FlowConnector

Connectors determine the order in which the nodes of the flow are executed. A connector defines and links to the subsequent node. It extends FlowBaseElement and inherits all its fields.

Field Name	Field Type	Description
`targetReference`	string	Required. Which node to execute after completing the current node.

FlowDataTypeMapping

This data type mapping defines the specific sObject data type for input and out values that have the generic sObject data type. It extends FlowBaseElement and inherits all its fields. Available in API version 48.0 and later.

Field Name	Field Type	Description
`typeName`	string	Required. API name of the input or output variable. The `T__` prefix is required for input variables. The `U__` prefix is required for output variables. For example, `T__inputCollection` represents the API name of the input variable `inputCollection`.
`typeValue`	string	Required. API name of the specific sObject data type that this value maps to. For example, `Account`.

FlowConstant

A constant resource defines a fixed value that can be used throughout your flow. It extends FlowElement and inherits all of its fields.

Field Name	Field Type	Description
`dataType`	FlowDataType (enumeration of type string)	Required. Valid types are: - `Currency` - `Date` - `Number` - `String` - `Boolean`
`value`	FlowElementReferenceOrValue	Default value of the constant. This field can't have merge fields, nor can it reference another resource besides `$GlobalConstant.EmptyString`.

FlowDecision

Decision node that evaluates a set of rules and routes the flow execution based on the first rule that evaluates to true. It extends FlowNode and inherits all its fields.

Field Name	Field Type	Description
`defaultConnector`	FlowConnector	Specifies which node to execute if none of the rules evaluate to true.
`defaultConnectorLabel`	string	Label for the default connector.
`rules`	FlowRule[]	An array of rules for the decision. The rules are evaluated in the order they're listed, and the connector of the first true rule is used. If no rules are true, then the default connector is used. In Flow Builder, rules are referred to as "decision outcomes."

FlowDynamicChoiceSet

Looks up data or metadata from an object and dynamically generates a set of choices at run time. It extends FlowElement and inherits all of its fields. Depending on the fields that are set, this element represents either a record choice or a picklist choice.

- A *record choice* dynamically generates choices based on records that meet specified filter criteria. If a dynamic choice doesn't have the `picklistField` and `picklistObject` parameters set, it is a record choice and can't have a data type of `Picklist` or `Multipicklist`.
- A *picklist choice* dynamically generates choices based on the available values for a picklist or multi-select picklist field. If a dynamic choice has the `picklistField` and `picklistObject` parameters set, it is a picklist choice and must have a data type of `Picklist` or `Multipicklist`.

Field Name	Field Type	Description
`dataType`	FlowDataType (enumeration of type string)	Required. Valid types are: - `Currency` - `Date`

Field Name	Field Type	Description
		- `Number` - `String` - `Boolean` - `Picklist`—Picklist choices only - `Multipicklist`—Picklist choices only `Picklist` and `Multipicklist` are available in API version 35.0 and later.
`displayField`	string	Required for record choices. Specifies the object field. The values of the object field are displayed to the user as choice labels for selecting a record. For example, for an account, specify `Name` in `displayField` if you want the dynamically generated choices to be displayed as the account names from the records that are retrieved from the database. Not supported for picklist choices. Picklist choices always display the labels for the retrieved picklist values.
`filters`	FlowRecordFilter[]	An array of filters to apply to the records that are retrieved from the database. For example, filter accounts to include only the accounts that were created in the past three months. Not supported for picklist choices.
`limit`	int	Maximum number of choices to include in the generated set of choices. Maximum and default: 200. If `sortField` and `sortOrder` are also specified, the records are sorted before the `limit` takes effect. This field is available in API version 25.0 and later. This field is nillable in API version 45.0 and later.
`object`	string	Required for record choices. The object whose fields you want to retrieve from the database and use to generate the set of choices. For example, use “Account” to dynamically generate choices from the information in account records in the database. Not supported for picklist choices.
`outputAssignments`	FlowOutputFieldAssignment[]	An array that assigns fields from the user-selected record to variables that can be used elsewhere in the flow. For example, when the user selects an account name from the dynamically generated list of choice options, outputAssignments can assign the Id and AnnualRevenue from the user-selected account to variables that you specify.

Field Name	Field Type	Description
		Not supported for picklist choices.
`picklistField`	string	Required for picklist choices. The field whose available values you want to retrieve from the database and use to generate the picklist choice. For example, use "Industry" to dynamically generate one choice for each available value on the Industry picklist field. Not supported for record choices. This field is available in API version 35.0 and later.
`picklistObject`	string	Required for picklist choices. The object whose field metadata you want to retrieve from the database and use to generate the picklist choice. For example, use "Account" to dynamically generate choices from a picklist field on the Account object. Not supported for record choices. This field is available in API version 35.0 and later.
`sortField`	string	Field that is used for sorting the records that meet the filter criteria. If this field isn't specified, then the returned records are not sorted. You can only sort records by fields that have the `Sort` API field property, as specified in SOAP API. Not supported for picklist choices. This field is available in API version 25.0 and later.
`sortOrder`	SortOrder (enumeration of type string)	Order in which to sort the records. If this field isn't specified, then the results are not sorted. Valid values are: • `Asc`—Ascending • `Desc`—Descending Not supported for picklist choices. This field is available in API version 25.0 and later.
`valueField`	string	Stored value for the choice, which can differ from what is displayed to the user as the choice options (`displayField`). For example, the `displayField` might be the account "Name" while the valueField is the account "Id." Not supported for picklist choices. Picklist choices always store the API value for the retrieved picklist values.

FlowElement

Base class for all flow elements. This is an abstract class. It extends FlowBaseElement and inherits all its fields.

Field Name	Field Type	Description
`description`	string	Description of the flow element.
`name`	string	Required. Unique name of the flow element.

FlowBaseElement

Base class for all flow elements that require contextual information in metadata values. This is an abstract class. FlowBaseElement is available in API version 32.0 and later.

Field Name	Field Type	Description
`processMetadataValues`	FlowMetadataValue[]	Contextual information for the element.

FlowMetadataValue

Defines contextual information that can be passed between elements in a flow. Flow metadata values can be used in an application that produces or consumes flows. FlowMetadataValue is available in API version 31.0 and later.

Field Name	Field Type	Description
`name`	string	Required. Name for the metadata value. This name doesn’t need to be unique across all elements.
`value`	FlowElementReferenceOrValue	Reference or value for the metadata value.

FlowElementReferenceOrValue

Defines a reference to an existing element or a particular value that you specify. Make sure that you specify only *one* of the fields.

Field Name	Field Type	Description
`booleanValue`	boolean	Use this field to specify a boolean value. Do not use this field if you want to specify a different data type or an element reference.
`dateTimeValue`	dateTime	Use this field to specify a dateTime value. Do not use this field if you want to specify a different data type or an element reference. This field is available in API version 30.0 and later.
`dateValue`	date	Use this field to specify a date value. Do not use this field if you want to specify a different data type or an element reference.
`elementReference`	string	Use this field to specify the name of an existing element. Do not use this field if you want to specify a value instead of an element reference.

Field Name	Field Type	Description
`numberValue`	double	Use this field to specify a double value. Do not use this field if you want to specify a different data type or an element reference.
`stringValue`	string	Use this field to specify a string value. Do not use this field if you want to specify a different data type or an element reference.

FlowFormula

Calculates a value using functions and elements in the flow. It extends FlowElement and inherits all its fields.

Field Name	Field Type	Description
`dataType`	FlowDataType (enumeration of type string)	The data type for the formula. Valid values are: • `Boolean` • `Currency` • `Date` • `DateTime` • `Number` • `String` `dataType` defaults to `Number` if it isn't defined in a formula. This field is available in API version 31.0 and later.
`expression`	string	Required. Salesforce formula expression. The return value must match the data type. See "Flow Formula Considerations" in Salesforce Help. For API version 30.0 and earlier, the return value must be numeric.
`scale`	int	Scale of the return value, specifically, the number of digits to the right of the decimal point. Available only when the data type is Number or Currency. Corresponds to the Decimal Places field in Flow Builder.

FlowInputFieldAssignment

Assigns the value for a record field based on a resource or static value. It extends FlowBaseElement and inherits all its fields.

Field Name	Field Type	Description
`field`	string	Required. Name of the field that is to be assigned a value while a record is being created or updated.
`value`	FlowElementReferenceOrValue	Value that is to be assigned to the field.

FlowInputValidationRule

Validation rules verify that the data entered by the user meets the specified requirements. If the validation rule evaluates to false, then the specified error message is displayed.

Field Name	Field Type	Description
`errorMessage`	string	Required. Error message to display when `formulaExpression` evaluates to false.
`formulaExpression`	string	Required. Boolean formula used to validate the user input. See "Flow Formula Considerations" in Salesforce Help.

FlowLoop

A construct for iterating through a collection. It extends FlowNode and inherits all of its fields. FlowLoop is available in API version 30.0 and later.

Field Name	Field Type	Description
`nextValueConnector`	FlowConnector	Points to the element that the flow navigates to for each of the entries in the collection. This is where the flow goes for the next value in the collection.
`noMoreValuesConnector`	FlowConnector	Points to the element to navigate to when all entries in the collection have been looped through.
`collectionReference`	string	Required. The collection being looped through.
`assignNextValueToReference`	string	Required. The variable to which the current value in the collection is assigned before navigating to the target of `nextValueConnector`.
`iterationOrder`	iterationOrder (enumeration of type string)	Valid values are: • `Asc`—Iterate through the collection in the order the values are listed (first to last). • `Desc`—Iterate through the collection in the reverse order the values are listed (last to first).

FlowNode

A node is a type of element that is visible in the flow diagram. It extends FlowElement and inherits all its fields.

Field Name	Field Type	Description
`label`	string	Required. Name of the node. This non-unique label is different from the unique name of the node, which is inherited from FlowElement.
`locationX`	int	Required. Horizontal location of the node, in pixels from the left.
`locationY`	int	Required. Vertical location of the node, in pixels from the top.

FlowOutputFieldAssignment

Assigns a record field's value from a record to a variable that can be used elsewhere in the flow. The record may be selected by a record lookup or via a user selection for a choice. It extends FlowBaseElement and inherits all its fields.

Field Name	Field Type	Description
`assignToReference`	string	Required. Reference to the variable where you want to store the value of the record field.
`field`	string	Required. Name of the field whose value is to be assigned after a record lookup.

FlowRecordCreate

Create a new record in the database using values from the flow. It extends FlowNode and inherits all its properties.

Note: The flow record create, lookup, update, and delete operations are different from the CRUD-based metadata calls `create()`, `retrieve()`, `update()`, and `delete()`. The flow record methods apply to record operations from within a flow, which aren't the same as doing any metadata calls to CRUD setup entities.

Field Name	Field Type	Description
`assignRecordIdToReference`	string	Reference to the variable where you want to store the ID after the record is created.
`connector`	FlowConnector	Specifies which node to execute after creating the record.
`faultConnector`	FlowConnector	Specifies which node to execute if the attempt to create a record results in an error.
`inputAssignments`	FlowInputFieldAssignment[]	An array that assigns values to the specified fields of the record being created.
`inputReference`	string	Specifies the record variable whose field values are used to populate the new record's fields.
`object`	string	Required. Object for the record to be created by this element
`storeOutputAutomatically`	boolean	Indicates whether the record ID is automatically available in the flow without creating any variables. When the value is `true`, you can reference the record ID by specifying the API name of the Create Records element in the flow. The default value is `false`. When the value is `false`, create a variable to store the record ID. This field is available in API version 48.0 and later.

FlowRecordDelete

Deletes one or more records in the database. It extends FlowNode and inherits all its fields.

Note: The flow record create, lookup, update, and delete operations are different from the CRUD-based metadata calls `create()`, `retrieve()`, `update()`, and `delete()`. The flow record methods apply to record operations from within a flow, which aren't the same as doing any metadata calls to CRUD setup entities.

Field Name	Field Type	Description
`connector`	FlowConnector	Specifies which node to execute after deleting the record.
`faultConnector`	FlowConnector	Specifies which node to execute if the attempt to delete a record results in an error.
`filters`	FlowRecordFilter[]	An array that specifies the criteria used to select which records to delete from the database. For example, delete accounts whose last activity was older than a specified date.
`object`	string	Required. The name of the object whose records are deleted.
`inputReference`	string	Specifies the record variable whose record ID is used to identify which record to delete in the database.

FlowRecordFilter

Sets the criteria for searching records in the database. It extends FlowBaseElement and inherits all its fields.

Field Name	Field Type	Description
`field`	string	Required. The field to be used for filtering records.
`operator`	FlowRecordFilterOperator (enumeration of type string)	Required. Valid values are: - `EqualTo` - `NotEqualTo` - `GreaterThan` - `LessThan` - `GreaterThanOrEqualTo` - `LessThanOrEqualTo` - `StartsWith` - `EndsWith` - `Contains` - `IsNull`
`value`	FlowElementReferenceOrValue	Reference or value used with the field and operator to filter records.

FlowRecordLookup

Finds records in the database and stores their field values in the flow. Corresponds to a Get Records element in Flow Builder. It extends FlowNode and inherits all of its fields.

Note: The flow record create, lookup, update, and delete operations are different from the CRUD-based metadata calls `create()`, `retrieve()`, `update()`, and `delete()`. The flow record methods apply to record operations from within a flow, which aren't the same as doing any metadata calls to CRUD setup entities.

Field Name	Field Type	Description
`assignNullValuesIfNoRecordFound`	boolean	Specifies that all values are set to `null` when no record is found. Supported only when `storeOutputAutomatically` is `false`. This field is available in API version 30.0 and later.
`connector`	FlowConnector	Specifies which node to execute after getting records from the database.
`faultConnector`	FlowConnector	Specifies which node to execute if the attempt to get records results in an error.
`filters`	FlowRecordFilter[]	An array that specifies the criteria used to select the record from the database. If the filters return more than one record, they are sorted according to the specified `sortField` and `sortOrder`. If `outputReference` specifies a non-collection record variable or if `getFirstRecordOnly` is `true`, only the first record in the sorted list is selected. If `sortField` or `sortOrder` is not specified, records are not returned in any particular order. If `outputReference` specifies a non-collection record variable or if `getFirstRecordOnly` is `true`, only the first record in the unsorted list is selected.
`getFirstRecordOnly`	boolean	Indicates whether to store field values for only one record, even when multiple records meet the filter criteria. Supported only when `storeOutputAutomatically` is `true`. When `storeOutputAutomatically` is `false`, what determines whether

Field Name	Field Type	Description
		one or multiple records are stored is whether `outputReference` specifies a record variable or a record collection variable. This field is available in API version 47.0 and later.
`limit`	int	The maximum number of records to return, to limit the amount of data received. This field is available in API version 30.0 and later.
`object`	string	Required. Name of the object from which to select the record.
`outputAssignments`	FlowOutputFieldAssignment[]	An array that assigns fields from the selected record to variables that can be used elsewhere in the flow. Supported only when `storeOutputAutomatically` is `false`.
`outputReference`	string	Specifies the record variable or record collection variable that stores the queried fields' values. Supported only when `storeOutputAutomatically` is `false`.
`queriedFields`	string[]	An array that specifies which fields from the selected record is saved to the specified record variable.
`sortField`	string	Field that is used for sorting the records that meet the filter criteria. If this field isn't specified, then the returned records are not sorted. You can only sort records by fields that have the `Sort` API field property, as specified in SOAP API. This field is available in API version 25.0 and later.
`sortOrder`	SortOrder (enumeration of type string)	Order in which to sort the records. If this field isn't specified, then the results are not sorted. Valid values are: • `Asc`—Ascending

Field Name	Field Type	Description
		- Desc—Descending This field is available in API version 25.0 and later.
storeOutputAutomatically	boolean	Indicates whether the returned records' field values are automatically available in the flow without creating any variables. When the value is true, the flow can reference a field by specifying the name of the Get Records element and the record field, such as Get_Contacts.AccountId. Supported only when processType is Flow or AutoLaunchedFlow. This field is available in API version 47.0 and later.

FlowRecordUpdate

Finds records in the database and updates them with values from the flow. It extends FlowNode and inherits all of its fields.

Note: The flow record create, lookup, update, and delete operations are different from the CRUD-based metadata calls `create()`, `retrieve()`, `update()`, and `delete()`. The flow record methods apply to record operations from within a flow, which aren't the same as doing any metadata calls to CRUD setup entities.

Field Name	Field Type	Description
connector	FlowConnector	Specifies which node to execute after completing the record update.
faultConnector	FlowConnector	Specifies which node to execute if the attempt to update a record results in an error.
filters	FlowRecordFilter[]	An array that specifies the criteria used to select the records to update in the database.
inputAssignments	FlowInputFieldAssignment[]	An array that assigns values to the specified fields of the record being updated.
object	string	Required. Name of the object whose records are updated.
inputReference	string	Specifies the record variable whose field values are used to update the record's fields.

FlowRule

Defines the conditions and logic that would enable a rule to evaluate to true. It extends FlowElement and inherits all of its fields.

Field Name	Field Type	Description
`conditionLogic`	string	Specifies logic for the conditions. Value can be: • `and`—evaluates to `true` only if all its conditions evaluate to true • `or`—evaluates to `true` if any of its conditions evaluate to true • Advanced logic like `1 AND (2 OR 3)`—evaluates to true if the first condition is true and either the second or third condition is true When you use advanced logic, the string must consist of 1,000 or fewer characters.
`conditions`	FlowCondition[]	An array of conditions for the rule.
`connector`	FlowConnector	Specifies which node to execute if this is the first rule that evaluates to true in a decision.
`label`	string	Required. Label for the connector.

FlowScreen

Screens capture information from users and display information to users. It extends FlowNode and inherits all its fields.

Field Name	Field Type	Description
`allowBack`	boolean	Indicates whether to show (true) or hide (false) the **Previous** button on the screen at runtime. When true, the **Previous** button appears only if the user visited a previous screen in the flow path and if `showFooter` for the screen is set to true. Set this to false when revisiting the previous screen would trigger an action that should not be repeated, such as a credit card transaction. This field is available in API version 26.0 and later. Default: true You can set either `allowBack` or `allowFinish` to false, but not both.
`allowFinish`	boolean	Indicates whether to show (true) or hide (false) the **Finish** button on the screen at runtime. When true, the **Finish** button appears only if the screen element is the end of a flow path and if `showFooter` for the screen is set to true. Set this to false if you need the user to go back to a previous screen to continue or complete the flow. For example, you wouldn't want to offer a **Finish**

Field Name	Field Type	Description
		button on a screen that tells the user to go back and make corrections on a previous screen. This field is available in API version 26.0 and later. Default: true You can set either `allowBack` or `allowFinish` to false, but not both.
`allowPause`	boolean	Indicates whether to show (true) or hide (false) the **Pause** button on the screen at runtime. The default value is true. A flow screen displays the **Pause** button if all the following conditions are true. • In the organization's process automation settings, **Let users pause flows** is enabled. • `allowPause` for the screen is set to true. • If the flow is embedded in a Visualforce page, the <flow:interview> component has its `showAllowPause` attribute set to true. • The `showFooter` field for the screen is set to true. This field is available in API version 33.0 and later.
`connector`	FlowConnector	Specifies which node to execute after the screen node.
`fields`	FlowScreenField[]	An array of fields to display on the screen.
`helpText`	string	Text that appears if the end user clicks a link for help text. Supports merge fields in API version 26.0 and later.
`pausedText`	string	Confirmation message that appears when an end-user clicks **Pause**. This field is available in API version 33.0 and later.
`rules`	FlowScreenRule[]	Reserved for future use.
`showFooter`	boolean	Indicates whether to show (true) or hide (false) the screen's footer at Lightning runtime. Classic runtime isn't supported. The default value is true. The footer includes navigation actions for the screen. If `showFooter` is hidden, use Lightning components on the screen to show navigation actions. This field is available in API version 42.0 and later.

Field Name	Field Type	Description
`showHeader`	boolean	Indicates whether to show (true) or hide (false) the screen's header at Lightning runtime. Classic runtime isn't supported. The default value is true. The header includes access to help text for the screen. If `showHeader` is hidden, use Lightning components on the screen to show help text. This field is available in API version 42.0 and later.

FlowScreenField

Represents a screen component. FlowScreenField extends FlowElement and inherits all its fields.

Field Name	Field Type	Description
`choiceReferences`	string[]	An array of references to FlowChoices or FlowDynamicChoiceSets. The resulting choice options appear in the order specified in this array, where the element at index 0 provides the top-most choice option. Supported for the following types of screen components: • RadioButtons • DropdownBox • MultiSelectCheckboxes • MultiSelectPicklist Multi-select checkboxes and multi-select picklist fields are available in API version 26.0 and later.
`dataType`	FlowDataType (enumeration of type string)	Data type of the screen component. Only supported for the InputField, RadioButtons, and DropdownBox types of screen components. Valid data types are: • `Boolean` • `Currency` • `Date` • `DateTime` • `Number` • `String` Boolean input fields, which appear as checkbox fields at runtime, are available in API version 26.0 and later.

Field Name	Field Type	Description
		Only the string data type is supported for multi-select checkboxes and multi-select picklist fields. Multi-select fields are available in API version 26.0 and later. Date/time input fields are available in API version 43.0 and later.
`dataTypeMappings`	FlowDataTypeMapping[]	Reserved for future use.
`defaultSelectedChoiceReference`	string	The name of the FlowChoice element to use as the default value for the screen component. Supported for the following types of screen components: • RadioButtons • DropdownBox • MultiSelectCheckboxes • MultiSelectPicklist For DropdownBox field types only, if `defaultSelectedChoiceReference` is empty or null, the reference at index 0 of `choiceReferences` is used as the default value. You can specify only one FlowChoice element as the default value for multi-select checkboxes and multi-select picklist fields. Multi-select fields are available in API version 26.0 and later.
`defaultValue`	FlowElementReferenceOrValue	The value that is used by default when the screen component requires users to provide input. Only supported for InputField, LargeTextArea, and PasswordField.
`extensionName`	string	The name of the Lightning component to display. This field is available in API version 42.0 and later.
`fieldText`	string	Field label that is displayed on the screen. Supports merge fields.
`fieldType`	FlowScreenFieldType (enumeration of type string)	Required. Valid values are: • `DisplayText` • `InputField` • `LargeTextArea` • `PasswordField` • `RadioButtons`

Field Name	Field Type	Description
		- `DropdownBox` - `MultiSelectCheckboxes`—available in API version 26.0 and later - `MultiSelectPicklist`—available in API version 26.0 and later - `ComponentInstance`—available in API version 42.0 and later - `ComponentChoice` and `ComponentInput`—available in API version 48.0 and later for the `Survey` `processType` value only At runtime, each multi-select field stores its field value as a concatenation of the user-selected choice values, separated by semicolons. Any semicolons in the selected choice values are removed when added to the multi-select field value.
`helpText`	string	Text that appears if the end user clicks the help icon (ℹ) for the screen component. Supports merge fields in API version 26.0 and later.
`inputParameters`	FlowScreenFieldInputParameter[]	An array of input parameters. Supported only when `fieldType` is `ComponentInstance`. This field is available in API version 42.0 and later.
`isRequired`	boolean	Indicates whether the user must select a choice or provide input. Not supported for DisplayText or boolean inputField.
`isVisible`	boolean	Reserved for future use.
`outputParameters`	FlowScreenFieldOutputParameter[]	An array of output parameters. Supported only when `fieldType` is `ComponentInstance` and when `storeOutputAutomatically` is `false`. This field is available in API version 42.0 and later.
`scale`	int	Controls the number of digits to the right of the decimal point up to 17 places. If you leave this field blank or set it to zero, only whole

Field Name	Field Type	Description
		numbers appear when your flow runs. Available only when the data type is Number or Currency. Corresponds to the Decimal Places field in Flow Builder.
storeOutputAutomatically	boolean	Indicates whether the screen component's output parameters are automatically available in the flow without creating any variables. When the value is true, you can reference an output parameter by specifying the name of the screen component and the output parameter, such as Mailing_Address.City. Supported only when fieldType is ComponentInstance. This field is available in API version 47.0 and later.
validationRule	FlowInputValidationRule	A rule used to validate the user input when the screen component is of type InputField, LargeTextArea, or PasswordField.
visibilityRule	FlowVisibilityRule	A rule based on conditions that is used to render or hide the screen component. This field is available in API version 47.0 and later.

FlowScreenFieldInputParameter

Defines an input parameter from the flow to the extension. It extends FlowBaseElement and inherits all its fields. FlowScreenFieldInputParameter is available in API version 42.0.

Field Name	Field Type	Description
name	string	Required. Unique name for the input parameter.
value	string	Defines the value of the input parameter.

FlowScreenFieldOutputParameter

Defines an output parameter from the extension to the flow. It extends FlowBaseElement and inherits all its fields. FlowScreenFieldOutputParameter is available in API version 42.0.

Field Name	Field Type	Description
assignToReference	string	Required. Specifies the variable to which you want to assign the output parameter value.
name	string	Required. Unique name for the output parameter.

FlowStage

A section of your flow that can be represented in the UI, such as with breadcrumbs. It extends FlowElement and inherits all its fields.

When an interview starts, any stages where isActive is true are added to the $Flow.ActiveStages global variable, which holds a collection of stages. Each stage's stageOrder determines the order they're added in. The stage with the lowest stageOrder is assigned to the $Flow.CurrentStage global variable.

Field Name	Field Type	Description
isActive	boolean	Indicates whether the stage is active by default.
label	string	A user-friendly label for this stage.
stageOrder	int	Indicates how the stage should be ordered against other stages. The stageOrder value must be unique within the flow.

FlowStart

Represents the flow's Start element, which specifies how the flow starts. In an autolaunched flow, the Start element can also define when and how frequently to run the flow and whether to run the flow for a set of records that meet filter criteria.

FlowStart extends FlowNode and inherits all its fields except name and label. Available in API version 47.0 and later.

Field Name	Field Type	Description
connector	FlowConnector	Specifies which element to execute first.
filters	FlowRecordFilter[]	An array of filters to apply when retrieving records from the database at the scheduled start time. For example, filter accounts to include only the records that haven't been updated in the last 4 weeks. Available only when triggerType is Scheduled.
object	string	The object whose records you want to retrieve from the database. A flow interview starts for each record that meets the filter conditions. Available only when triggerType is Scheduled.
recordTriggerType	RecordTriggerType (enumeration of type string)	Specifies what type of record changes can start the flow: • Create—when a record is created • Update—when a record is updated • CreateAndUpdate—when a record is created and updated Available only when triggerType is RecordBeforeSave. This field is available in API version 48.0 and later.

Field Name	Field Type	Description
schedule	FlowSchedule	Required when triggerType is Scheduled. Specifies when and how frequently the flow runs.
triggerType	FlowTriggerType (enumeration of type string)	Specifies what causes the flow to run. If you exclude this field, the flow has no trigger and starts only when a user or app launches the flow. Valid value is: • Scheduled—The flow starts at a scheduled time. • RecordBeforeSave—Creating and/or updating a record triggers an autolaunched flow to make additional updates to that record before it's saved to the database. This value is available in API version 48.0 and later. Available only when processType is AutoLaunchedFlow.

FlowSchedule

Specifies when and how frequently to run the flow. Available in API version 47.0 and later.

Field Name	Field Type	Description
frequency	FlowStartFrequency (enumeration of type string)	Specifies how frequently to run the flow. Valid values are: • Once • Daily • Weekly
startDate	date	The date when the flow runs, or when the flow's run schedule starts recurring.
startTime	time	The time of day when the flow runs, based on the org's default time zone.

FlowStep

Steps function as placeholders when you're building a flow. It extends FlowNode and inherits all its fields.

Field Name	Field Type	Description
connectors	FlowConnector[]	Specifies which node to execute after the step node.

FlowSubflow

A subflow element references another flow, which it calls at run time. The flow that contains the subflow element is referred to as the master flow. FlowSubflow extends FlowNode and inherits all of its fields. It is available in API version 25.0 and later.

Field Name	Field Type	Description
connector	FlowConnector	Specifies which node to execute after the subflow.
flowName	string	References the flow to call at runtime. The value must be an API name of a flow and can't contain an appended hyphen and version number.
inputAssignments	FlowSubflowInputAssignment[]	An array of input variable assignments that are set at the start of the referenced flow.
outputAssignments	FlowSubflowOutputAssignment[]	An array of output variable assignments that are set at the end of the referenced flow.

FlowSubflowInputAssignment

Assigns an element or value from the master flow to a variable in the referenced flow. Input assignments occur when the subflow calls the referenced flow. It extends FlowBaseElement and inherits all its fields. It is available in API version 25.0 and later.

Field Name	Field Type	Description
name	string	Required. Unique name for the variable in the referenced flow.
value	FlowElementReferenceOrValue	Defines the value to assign to the variable.

FlowSubflowOutputAssignment

Assigns the value of a variable from the referenced flow to a variable in the master flow. Output assignments occur when the referenced flow is finished running. It extends FlowBaseElement and inherits all its fields. It is available in API version 25.0 and later.

Field Name	Field Type	Description
assignToReference	string	Required. Unique name for the variable in the master flow.
name	string	Required. Unique name for the variable in the referenced flow.

FlowTextTemplate

Defines a text template that can be used throughout the flow. It extends FlowElement and inherits all its fields.

Field Name	Field Type	Description
text	string	Actual text of the template. Supports merge fields.

FlowVariable

Variables allow you to create updatable values to use in the flow. FlowVariable extends FlowElement and inherits all its fields.

Field Name	Field Type	Description
`apexClass`	string	The Apex class of this variable if its data type is `Apex`. Available in API version 46.0 and later.
`dataType`	FlowDataType (enumeration of type string)	Required. Valid types are: - `Apex`—This value is available in API version 46.0 and later. - `Boolean` - `Currency` - `Date` - `DateTime`—This value is available in API version 30.0 and later. - `Number` - `Multipicklist`—This value is available in API version 34.0 and later. - `Picklist`—This value is available in API version 34.0 and later. - `String` - `sObject`—This value corresponds to a record variable.
`isCollection`	boolean	Indicates whether the variable is a collection of values. This field is available in API version 30.0 and later. In API version 32.0 and later, a collection variable can be of any data type. Default value is `False`.
`isInput`	boolean	Indicates whether the variable can be set at the start of the flow using URL parameters, Visualforce controllers, or subflow inputs. This field is available in API version 25.0 and later. Default value: - `False` for a variable created in API version 25.0 and later or in the Flow Builder in Summer '12 and later. - `True` for a variable created in API version 24.0 or in Flow Builder in Spring '12 and earlier. Warning: Disabling input or output access for an existing variable can break the functionality of applications and pages that call the

Field Name	Field Type	Description
		flow and access the variable. For example, you can access variables from URL parameters, processes, and other flows.
isOutput	boolean	Indicates whether the variable's value can be accessed from Visualforce controllers and other flows. This field is available in API version 25.0 and later. Default value: • False for a variable created in API version 25.0 and later or in the Flow Builder in Summer '12 and later. • True for a variable created in API version 24.0 or in Flow Builder in Spring '12 and earlier. Warning: Disabling input or output access for an existing variable can break the functionality of applications and pages that call the flow and access the variable. For example, you can access variables from URL parameters, processes, and other flows.
objectType	string	Object type of this variable if its data type is sObject.
scale	int	Controls the number of digits to the right of the decimal point up to 17 places. If you leave this field blank or set it to zero, only whole numbers appear when your flow runs. Available only when the data type is Number or Currency. Corresponds to the Decimal Places field in Flow Builder.
value	FlowElementReferenceOrValue	Default value of this variable. Default values aren't supported if the variable's data type is Picklist or Multipicklist.

FlowVisibilityRule

Visibility rules render a flow screen component when visibility rule conditions are met. Hides a flow screen component when visibility rule conditions aren't met. Available in API version 47.0 and later.

Field Name	Field Type	Description
`conditionLogic`	string	Specifies logic for the conditions. Value can be : - `and`—evaluates to `true` only if all its conditions evaluate to true - `or`—evaluates to `true` if any of its conditions evaluate to true - Advanced logic like `1 AND (2 OR 3)`—evaluates to `true` if the first condition is true and either the second or third condition is true When you use advanced logic, the string must consist of 1,000 or fewer characters.
`conditions`	FlowCondition[]	An array of conditions that must be true for the flow to wait for this event.

FlowWait

Waits for one or more defined events to occur. FlowWait extends FlowNode and inherits all its fields. FlowWait is available in API version 32.0 and later.

Field Name	Field Type	Description
`defaultConnector`	FlowConnector	Specifies which node to execute if the conditions are false for every event in the Wait element.
`defaultConnectorLabel`	string	Label for the default connector.
`faultConnector`	FlowConnector	Specifies which node to execute if the attempt to wait results in an error. If any of the wait events fail, the flow takes the fault connector.
`waitEvents`	FlowWaitEvent[]	An array of events that the Wait element is waiting for. If the conditions for every event evaluate to `false`, the `defaultConnector` is used.

FlowWaitEvent

An event that a FlowWait element is waiting for. FlowWaitEvent extends FlowElement and inherits all its fields. FlowWaitEvent is available in API version 32.0 and later.

Field Name	Field Type	Description
`conditionLogic`	string	Specifies logic for the conditions. Value can be: • `and`—evaluates to `true` only if all its conditions evaluate to true • `or`—evaluates to `true` if any of its conditions evaluate to true • Advanced logic like `1 AND (2 OR 3)`—evaluates to true if the first condition is true and either the second or third condition is true When you use advanced logic, the string must consist of 1,000 or fewer characters.
`conditions`	FlowCondition[]	An array of conditions that must be `true` for the flow to wait for this event.
`connector`	FlowConnector	Specifies which node to execute if this is the first event that occurs.
`eventType`	string	Required. The event's type. The type determines which input parameters are available to define this event. Valid values are: • `AlarmEvent`—This event is an alarm based off an absolute date/time value. • `DateRefAlarmEvent`—This event is an alarm based off a date/time field on a record.
`inputParameters`	FlowWaitEventInputParameter[]	An array of the event's input parameters. The parameter values are set by using values from the flow.
`label`	string	Required. Label for the wait event.
`outputParameters`	FlowWaitEventOutputParameter[]	An array of the event's output parameters. The parameter values are assigned from the event to variables in the flow.

FlowWaitEventInputParameter

An input parameter for FlowWaitEvent. The parameter's value is set by using values from the flow. It extends FlowBaseElement and inherits all its fields. FlowWaitEventInputParameter is available in API version 32.0 and later.

Field Name	Field Type	Description
name	string	Required. Unique name for the input parameter.
value	FlowElementReferenceOrValue	Defines the value of the input parameter.

FlowWaitEventOutputParameter

An output parameter for FlowWaitEvent. The parameter's value is assigned to a variable in the flow so that it can be referenced in another part of the flow. It extends FlowBaseElement and inherits all its fields. FlowWaitEventOutputParameter is available in API version 32.0 and later.

Field Name	Field Type	Description
assignToReference	string	Required. Specifies the variable to which you want to assign the output parameter value.
name	string	Required. Unique name for the output parameter.

Upgrade Flow Files to API Version 44.0 or Later

In API version 43.0 and earlier, the Flow object's fullName field included the flow's version number. Starting in API version 44, the field no longer includes the version number. Before you deploy using API version 44.0 via Metadata API or Salesforce CLI, make sure that:

- The `flows` directory doesn't include any unused flow versions.
- For each active flow, the `status` field is `Active`. Any flow without a `status` value is deployed or retrieved with a `status` value of `Draft`.
- The `flowDefinitions` directory is empty.

For Metadata API only.

- The `package.xml` file is set to API version 44.0.
- For the latest version of each flow, the file name doesn't include a version number. For example, change `myflow-3.flow` to `myflow.flow`.

For Salesforce CLI only.

- The `sfdx-project.json` file is set to `"sourceApiVersion": "44.0"`.
- For the latest version of each flow, the file name doesn't include a version number. For example, change `myflow-1.flow-meta.xml` to `myflow.flow-meta.xml`.

As part of this upgrade, flow definitions are no longer necessary when you deploy or retrieve via Metadata API. If you deploy with flow definitions, the active version numbers in the flow definitions override the `status` fields in the flows. For example, the active version number in the flow definition is version 3, and the latest version of the flow is version 4 with the `status` field as `Active`. After you deploy your flow, the active version is version 3.

After you finished this upgrade, you can integrate with a version control system without worrying about flow file names changing. To reduce deployment issues when you push the source code into a scratch org, make sure you don't reuse an existing scratch org.

Declarative Metadata Sample Definition

A sample XML definition of a flow is shown below.

```
<?xml version="1.0" encoding="UTF-8"?>
<Flow xmlns="http://soap.sforce.com/2006/04/metadata">
    <actionCalls>
        <name>Post_to_Contact_s_Feed</name>
        <label>Post to Contact&apos;s Feed</label>
        <locationX>269</locationX>
        <locationY>396</locationY>
        <actionName>chatterPost</actionName>
        <actionType>chatterPost</actionType>
        <connector>
            <targetReference>Confirm</targetReference>
        </connector>
        <inputParameters>
            <name>text</name>
            <value>
                <elementReference>chatterMessage</elementReference>
            </value>
        </inputParameters>
        <inputParameters>
            <name>subjectNameOrId</name>
            <value>
                <elementReference>contact.Id</elementReference>
            </value>
        </inputParameters>
        <storeOutputAutomatically>true</storeOutputAutomatically>
    </actionCalls>
    <actionCalls>
        <name>Get_Info</name>
        <label>Get Info</label>
        <locationX>372</locationX>
        <locationY>354</locationY>
        <actionName>GetFirstFromCollection</actionName>
        <actionType>apex</actionType>
        <dataTypeMappings>
            <typeName>T__inputCollection</typeName>
            <typeValue>Account</typeValue>
        </dataTypeMappings>
        <dataTypeMappings>
            <typeName>U__outputMember</typeName>
            <typeValue>Account</typeValue>
        </dataTypeMappings>
        <inputParameters>
            <name>inputCollection</name>
            <value>
                <elementReference>accts</elementReference>
            </value>
        </inputParameters>
        <storeOutputAutomatically>true</storeOutputAutomatically>
    <assignments>
        <name>Set_Contact_ID</name>
        <label>Set Contact ID</label>
        <locationX>568</locationX>
        <locationY>396</locationY>
        <assignmentItems>
            <assignToReference>contact.Id</assignToReference>
            <operator>Assign</operator>
            <value>
                <elementReference>existingId</elementReference>
            </value>
        </assignmentItems>
        <connector>
            <targetReference>Update_Contact</targetReference>
        </connector>
    </assignments>
    <decisions>
        <name>Update_If_Existing</name>
        <label>Update If Existing?</label>
        <locationX>260</locationX>
        <locationY>52</locationY>
        <defaultConnector>
            <targetReference>Create_Contact</targetReference>
        </defaultConnector>
        <defaultConnectorLabel>No</defaultConnectorLabel>
        <rules>
            <name>Update_Yes</name>
            <conditionLogic>and</conditionLogic>
            <conditions>
                <leftValueReference>updateExisting</leftValueReference>
                <operator>EqualTo</operator>
                <rightValue>
                    <booleanValue>true</booleanValue>
                </rightValue>
            </conditions>
            <connector>
                <targetReference>Find_a_Match</targetReference>
            </connector>
            <label>Yes</label>
        </rules>
    </decisions>
    <decisions>
        <name>Update_or_Create</name>
        <label>Update or Create?</label>
        <locationX>447</locationX>
        <locationY>239</locationY>
        <defaultConnector>
            <targetReference>Create_Contact</targetReference>
        </defaultConnector>
        <defaultConnectorLabel>Create New</defaultConnectorLabel>
        <rules>
            <name>Update_Existing</name>
            <conditionLogic>and</conditionLogic>
            <conditions>
                <leftValueReference>existingId</leftValueReference>
                <operator>IsNull</operator>
                <rightValue>
```

```
                    <booleanValue>false</booleanValue>
                </rightValue>
            </conditions>
            <connector>
                <targetReference>Set_Contact_ID</targetReference>
            </connector>
            <label>Update Existing</label>
        </rules>
    </decisions>
    <dynamicChoiceSets>
        <name>accounts</name>
        <dataType>String</dataType>
        <displayField>Name</displayField>
        <object>Account</object>
        <outputAssignments>
            <assignToReference>contact.AccountId</assignToReference>
            <field>Id</field>
        </outputAssignments>
        <valueField>Id</valueField>
    </dynamicChoiceSets>
    <formulas>
        <name>created_or_updated</name>
        <dataType>String</dataType>
        <expression>IF({!Create_Contact}, &quot;created&quot;,
&quot;updated&quot;)</expression>
    </formulas>
    <interviewLabel>New Contact {!$Flow.CurrentDateTime}</interviewLabel>
    <isAdditionalPermissionRequiredToRun>true</isAdditionalPermissionRequiredToRun>
    <isTemplate>true</isTemplate>
    <label>New Contact</label>
    <processMetadataValues>
        <name>BuilderType</name>
        <value>
            <stringValue>LightningFlowBuilder</stringValue>
        </value>
    </processMetadataValues>
    <processMetadataValues>
        <name>OriginBuilderType</name>
        <value>
            <stringValue>LightningFlowBuilder</stringValue>
        </value>
    </processMetadataValues>
    <processType>Flow</processType>
    <recordCreates>
        <name>Create_Contact</name>
        <label>Create Contact</label>
        <locationX>269</locationX>
        <locationY>241</locationY>
        <connector>
            <targetReference>Post_to_Contact_s_Feed</targetReference>
        </connector>
        <inputReference>contact</inputReference>
        <storeOutputAutomatically>true</storeOutputAutomatically>
    </recordCreates>
```

```
   <recordLookups>
       <name>Find_a_Match</name>
       <label>Find a Match</label>
       <locationX>456</locationX>
       <locationY>53</locationY>
       <assignNullValuesIfNoRecordsFound>true</assignNullValuesIfNoRecordsFound>
       <connector>
           <targetReference>Update_or_Create</targetReference>
       </connector>
       <filters>
           <field>FirstName</field>
           <operator>EqualTo</operator>
           <value>
               <elementReference>contact.FirstName</elementReference>
           </value>
       </filters>
       <filters>
           <field>LastName</field>
           <operator>EqualTo</operator>
           <value>
               <elementReference>contact.LastName</elementReference>
           </value>
       </filters>
       <object>Contact</object>
       <outputAssignments>
           <assignToReference>existingId</assignToReference>
           <field>Id</field>
       </outputAssignments>
       <storeOutputAutomatically>false</storeOutputAutomatically>
   </recordLookups>
   <recordUpdates>
       <name>Update_Contact</name>
       <label>Update Contact</label>
       <locationX>456</locationX>
       <locationY>396</locationY>
       <connector>
           <targetReference>Post_to_Contact_s_Feed</targetReference>
       </connector>
       <inputReference>contact</inputReference>
   </recordUpdates>
   <screens>
       <name>Confirm</name>
       <label>Confirm</label>
       <locationX>270</locationX>
       <locationY>519</locationY>
       <allowBack>false</allowBack>
       <allowFinish>true</allowFinish>
       <allowPause>true</allowPause>
       <fields>
           <name>confirmation_message</name>
         <fieldText>Thanks! &lt;a href=&quot;/{!contact.Id}&quot;&gt;The contact&lt;/a&gt;
was {!created_or_updated}.</fieldText>
           <fieldType>DisplayText</fieldType>
       </fields>
```

```
        <showFooter>true</showFooter>
        <showHeader>true</showHeader>
    </screens>
    <screens>
        <name>Contact_Info</name>
        <label>Contact Info</label>
        <locationX>160</locationX>
        <locationY>50</locationY>
        <allowBack>true</allowBack>
        <allowFinish>true</allowFinish>
        <allowPause>true</allowPause>
        <connector>
            <targetReference>Update_If_Existing</targetReference>
        </connector>
        <fields>
            <name>contactName</name>
            <extensionName>flowruntime:name</extensionName>
            <fieldType>ComponentInstance</fieldType>
            <isRequired>true</isRequired>
            <outputParameters>
                <assignToReference>contact.FirstName</assignToReference>
                <name>firstName</name>
            </outputParameters>
            <outputParameters>
                <assignToReference>contact.LastName</assignToReference>
                <name>lastName</name>
            </outputParameters>
        </fields>
        <fields>
            <name>Account</name>
            <choiceReferences>accounts</choiceReferences>
            <dataType>String</dataType>
            <fieldText>Account</fieldText>
            <fieldType>DropdownBox</fieldType>
            <isRequired>true</isRequired>
        </fields>
        <fields>
            <name>update_toggle</name>
            <extensionName>flowruntime:toggle</extensionName>
            <fieldType>ComponentInstance</fieldType>
            <inputParameters>
                <name>label</name>
                <value>
                    <stringValue>If this contact already exists, update the existing
record.</stringValue>
                </value>
            </inputParameters>
            <inputParameters>
                <name>messageToggleActive</name>
                <value>
                    <stringValue>Update existing</stringValue>
                </value>
            </inputParameters>
            <inputParameters>
```

```
                <name>messageToggleInactive</name>
                <value>
                    <stringValue>Create other contact</stringValue>
                </value>
            </inputParameters>
            <isRequired>true</isRequired>
            <outputParameters>
                <assignToReference>updateExisting</assignToReference>
                <name>value</name>
            </outputParameters>
        </fields>
        <showFooter>true</showFooter>
        <showHeader>true</showHeader>
    </screens>
    <start>
        <locationX>50</locationX>
        <locationY>50</locationY>
        <connector>
            <targetReference>Contact_Info</targetReference>
        </connector>
    </start>
    <status>Draft</status>
    <textTemplates>
        <name>chatterMessage</name>
        <text>The contact was {!created_or_updated}.</text>
    </textTemplates>
    <variables>
        <name>contact</name>
        <dataType>SObject</dataType>
        <isCollection>false</isCollection>
        <isInput>false</isInput>
        <isOutput>false</isOutput>
        <objectType>Contact</objectType>
    </variables>
    <variables>
        <name>existingId</name>
        <dataType>String</dataType>
        <isCollection>false</isCollection>
        <isInput>false</isInput>
        <isOutput>false</isOutput>
    </variables>
    <variables>
        <name>updateExisting</name>
        <dataType>Boolean</dataType>
        <isCollection>false</isCollection>
        <isInput>false</isInput>
        <isOutput>false</isOutput>
    </variables>
</Flow>
```

Wildcard Support in the Manifest File

This metadata type supports the wildcard character `*` (asterisk) in the `package.xml` manifest file. For information about using the manifest file, see Deploying and Retrieving Metadata with the Zip File.

FlowCategory

Represents a list of flows that are grouped by category. Flows aren't added directly to a Lightning Bolt Solution. Instead, add the category the flows are in to the Lightning Bolt Solution. This type extends the Metadata metadata type and inherits its `fullName` field.

File Suffix and Directory Location

FlowCategory components have the suffix `.flowCategory` and are stored in the `flowCategories` folder.

Version

FlowCategory components are available in API version 43.0 and later.

Fields

Field Name	Field Type	Description
`description`	string	The description of this flow category.
`flowCategoryItems`	FlowCategoryItems[]	The list of flows in this flow category.
`masterLabel`	string	Required. The label for this flow category, which appears in Setup.

FlowCategoryItems

Represents the list of flows in a flow category.

Field Name	Field Type	Description
`flow`	string	Required. The name of the flow.

Declarative Metadata Sample Definition

The following is an example of a FlowCategory component.

```
<?xml version="1.0" encoding="UTF-8"?>
<FlowCategory xmlns="http://soap.sforce.com/2006/04/metadata"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
    <flowCategoryItems>
        <flow>PausableFlow</flow>
    </flowCategoryItems>
    <flowCategoryItems>
```

```
        <flow>BankingFlow</flow>
    </flowCategoryItems>
    <masterLabel>updateBenefits</masterLabel>
        <description>All the update benefits.</description>
</FlowCategory>
```

The following is an example `package.xml` that references the previous definition.

```
<?xml version="1.0" encoding="UTF-8"?>
<Package xmlns="http://soap.sforce.com/2006/04/metadata">
    <types>
        <members>updateBenefits</members>
        <name>FlowCategory</name>
    </types>
    <version>43.0</version>
</Package>
```

Wildcard Support in the Manifest File

This metadata type supports the wildcard character `*` (asterisk) in the `package.xml` manifest file. For information about using the manifest file, see Deploying and Retrieving Metadata with the Zip File.

FlowDefinition

Represents the flow definition's description and active flow version number.

Important: In API version 44.0, we recommend upgrading your flows to flow metadata file names without version numbers and discontinue using the FlowDefinition object to activate or deactivate a flow. Then use the Flow object to activate or deactivate a flow. For more information, see Upgrade Flow Files to API Version 44.0 on page 569.

If you deploy with flow definitions, the active version numbers in the flow definitions override the `status` fields in the flows. For example, the active version number in the flow definition is version 3, and the latest version of the flow is version 4 with the `status` field as `Active`. After you deploy your flow, the active version is version 3.

Declarative Metadata File Suffix and Directory Location

FlowDefinitions are stored in the `flowDefinitions` directory of the corresponding package directory. The file name matches the flow definition's unique full name, and the extension is `.flowDefinition`.

Version

FlowDefinition is available in API version 34.0 and later.

Field Name	Field Type	Description
`activeVersionNumber`	int	The version number of the active flow.
`description`	string	Description of the flow definition.

Field Name	Field Type	Description
`masterLabel`	string	Label for the flow definition. In managed packages, this field inherits the flow's active version name. To change this label from a subscriber's org, edit the packaged flow name.

Wildcard Support in the Manifest File

This metadata type supports the wildcard character `*` (asterisk) in the `package.xml` manifest file. For information about using the manifest file, see Deploying and Retrieving Metadata with the Zip File.

Folder

Represents a folder. This type extends the Metadata metadata type and inherits its `fullName` field.

Four folder types currently exist in Salesforce:

- Document folder
- Email folder (available for Salesforce Classic email templates only)
- Report folder
- Dashboard folder

Folder type names end with the "Folder" suffix. For example, the type name of an email folder is "EmailFolder".

File Suffix and Directory Location

Folders are stored in the corresponding component directory of the package. These directories are named `documents`, `email`, `reports`, and `dashboards`. Folders do not have a text file representation—they are containers for files. For each folder, an accompanying metadata file named *`FolderName`*`-meta.xml` is created at the same directory level. The *`FolderName`*`-meta.xml` metadata file contains the metadata information for that folder, such as the `accessType`. For example, for a documents folder named sampleFolder, there's a *`sampleFolder-meta.xml`* within the `documents` folder of the package.

Version

Folders are available in API version 11.0 and later.

Fields

This metadata type contains the following fields:

Field Name	Field Type	Description
`accessType`	FolderAccessTypes (enumeration of type string)	Required. The type of access for this folder. Valid values are: • `Shared`. This folder is accessible only by the specified set of users. • `Public`. This folder is accessible by all users, including portal users.

Field Name	Field Type	Description
		• PublicInternal. This folder is accessible by all users, excluding portal users. This setting is available for report and dashboard folders in organizations with a partner portal or Customer Portal enabled. • Hidden. This folder is hidden from all users.
fullName	string	The name used as a unique identifier for API access. The fullName can contain only underscores and alphanumeric characters. It must be unique, begin with a letter, not include spaces, not end with an underscore, and not contain two consecutive underscores. This field is inherited from the Metadata component.
name	string	Required. The name of the document folder.
publicFolderAccess	PublicFolderAccess (enumeration of type string)	If Public is the value for accessType, this field indicates the type of access all users have to the contents of the folder. Valid values include: • ReadOnly. All users can read the contents of the folder, but no user can change the contents. • ReadWrite. All users can read or change the contents of the folder.
sharedTo	SharedTo	Sharing access for the folder. See “Sharing Considerations” in Salesforce Help.

Declarative Metadata Sample Definition

The following is the package manifest definition of a document folder that contains a document:

```
<?xml version="1.0" encoding="UTF-8"?>
<Package xmlns="http://soap.sforce.com/2006/04/metadata">
    <fullName>basic</fullName>
    <types>
        <members>sampleFolder</members>
        <members>sampleFolder/TestDocument.txt</members>
        <name>Document</name>
    </types>
    <version>48.0</version>
</Package>
```

The following is an example of the sampleFolder-meta.xml metadata file for the sampleFolder document folder:

```
<?xml version="1.0" encoding="UTF-8"?>
<DocumentFolder xmlns="http://soap.sforce.com/2006/04/metadata">
    <accessType>Public</accessType>
    <name>sampleFolder</name>
    <publicFolderAccess>ReadWrite</publicFolderAccess>
</DocumentFolder>
```

Wildcard Support in the Manifest File

This metadata type doesn't support the wildcard character `*` (asterisk) in the `package.xml` manifest file. For information about using the manifest file, see Deploying and Retrieving Metadata with the Zip File.

IN THIS SECTION:

1. FolderShare

SEE ALSO:

Dashboard

Document

EmailTemplate

Report

FolderShare

Represents the settings for enhanced analytics folder sharing. Users can control access to reports or dashboards by giving others Viewer, Editor or Manager access to the folder that contains the report or dashboard.

File Suffix and Directory Location

FolderShare objects are stored in the `reports` and `dashboards` directories. For each report or dashboard folder it contains, there is a metadata file named *`FolderName`*`-meta.xml`. The *`FolderName`*`-meta.xml` metadata file contains the metadata information for that folder, such as the `accessLevel`. For example, if the `reports` directory contains a reports folder named `myReportsFolder`, it also has a *`myReportsFolder-meta.xml`* file at the same level as `myReportsFolder`.

Version

FolderShare components are available in API version 28 and later.

Fields

Field Name	Field Type	Description
`accessLevel`	FolderShareAccessLevel (enumeration of type string)	Required. Specifies the combination of actions that can be taken on the folder. Valid values are: • `View`. User can run a report or refresh a dashboard, but can't edit them. All users have at least Viewer access to report and dashboard folders that have been shared with them. (Some users may have administrative permissions that give them greater access.) • `EditAllContents`. Users can view and modify the reports or dashboards in the folder, and move them to and from any other folders that they have equivalent access to. • `Manage`. Users can do everything Viewers and Editors can do, plus control other users' access to a folder.

Field Name	Field Type	Description
`sharedTo`	string	Required. Specifies the user, group, or role that has the specified access level to the folder.
`sharedToType`	FolderSharedToType(enumeration of type string)	Required. Specifies the type of entity that the folder is shared with. Valid values are: • `Group`. Users in a specified public group have the specified access level to the folder. • `Manager`. Available in API version 29.0 and later. • `ManagerAndSubordinatesInternal`. Available in API version 29.0 and later. • `Role`. Users with a specified role have the specified access level to the folder. • `RoleAndSubordinates`. Users with a specified role, and users with a role subordinate to that role, have the specified access level to the folder. • `RoleAndSubordinatesInternal`. Users with a specified role and users with a role subordinate to that role, except public portal users, have the specified access level to the folder. • `Organization`. All internal users have the specified access level to the folder. • `Territory`. Users in a specified territory have the specified access level to the folder. • `TerritoryAndSubordinates`. Users in a specified territory, and users in territories subordinate to that, have the specified access level to the folder. • `AllPrmUsers`. All PRM Portal users have the specified level of access to the folder. • `User`. The specified individual user has the specified level of access to the folder. • `PartnerUser`. The specified individual user of a partner portal has the specified level of access to the folder. • `AllCspUsers`. All Customer Success Portal users have the specified level of access to the folder. • `CustomerPortalUser`. The specified individual user of a customer portal has the specified level of access to the folder. • `PortalRole`. Users with a specified role in a portal have the specified access level to the folder. • `PortalRoleAndSubordinates`. Portal users with a specified role, and portal users with a role subordinate to that role, have the specified access level to the folder.

Declarative Metadata Sample Definition

The following is an example of a FolderShare component for a dashboard folder:

```
<?xml version="1.0" encoding="UTF-8"?>
<DashboardFolder xmlns="http://soap.sforce.com/2006/04/metadata">
  <folderShares>
        <accessLevel>View</accessLevel>
        <sharedTo>R1</sharedTo>
        <sharedToType>Role</sharedToType>
    </folderShares>
</DashboardFolder>
```

This is an example of a FolderShare component for a report folder:

```
<?xml version="1.0" encoding="UTF-8"?>
<ReportFolder xmlns="http://soap.sforce.com/2006/04/metadata">
  <folderShares>
       <accessLevel>View</accessLevel>
        <sharedTo>R1</sharedTo>
        <sharedToType>Role</sharedToType>
    </folderShares>
</ReportFolder>
```

Wildcard Support in the Manifest File

This metadata type doesn't support the wildcard character * (asterisk) in the package.xml manifest file. For information about using the manifest file, see Deploying and Retrieving Metadata with the Zip File.

GlobalPicklist

Represents a global picklist, or the set of shared picklist values that custom picklist fields can use. (In contrast, the custom picklist fields that are based on a global picklist are of type CustomValue.) This type extends the Metadata metadata type and inherits its fullName field.

File Suffix and Directory Location

GlobalPicklist components have the suffix .globalPicklist and are stored in the globalPicklist folder.

Version

GlobalPicklist components are available in API version 37.0 only. In API version 38.0 and later, GlobalPicklist is replaced by the GlobalValueSet on page 586 type.

Fields

Field Name	Field Type	Description
description	string	It's useful to state the global picklist's purpose, and which objects it's intended for. Limit: 255 characters.

Field Name	Field Type	Description
`globalPicklistValues`	GlobalPicklistValue on page 584[]	Requires at least one value. The list of values (or "picklist value set") that's defined for a global picklist. The picklist value set is inherited by any custom picklist field that's based on that global picklist. Each value is of type GlobalPicklistValue. A global picklist can have up to 1,000 total values (inclusive of inactive values).
`masterLabel`	string	Required. A global picklist's name, which is defined when the global picklist is created. Appears as Label in the user interface.
`sorted`	string	Indicates whether a global picklist's value set is sorted in alphabetical order. By default this value is `false`.

Declarative Metadata Sample Definition

The following Territories.globalPicklist is an example of a GlobalPicklist component.

```
<?xml version="1.0" encoding="UTF-8"?>
<GlobalPicklist xmlns="http://soap.sforce.com/2006/04/metadata">
    <description>Updated:This is a basic global picklist</description>
    <globalPicklistValues>
        <fullName>Northwest</fullName>
        <default>false</default>
    </globalPicklistValues>
    <globalPicklistValues>
        <fullName>Northeast</fullName>
        <default>false</default>
    </globalPicklistValues>
    <globalPicklistValues>
        <fullName>South</fullName>
        <default>true</default>
    </globalPicklistValues>
    <globalPicklistValues>
        <fullName>Southwest</fullName>
        <default>false</default>
        <isActive>false</isActive>
    </globalPicklistValues>
    <masterLabel>Territories</masterLabel>
    <sorted>true</sorted>
</GlobalPicklist>
```

The following is an example `package.xml` that references the previous definition.

```
<?xml version="1.0" encoding="UTF-8"?>
<Package xmlns="http://soap.sforce.com/2006/04/metadata">
    <types>
        <members>Territories</members>
        <name>GlobalPicklist</name>
    </types>
    <version>37.0</version>
</Package>
```

GlobalPicklistValue

Represents the definition of a value used in a global picklist. Custom picklist fields can inherit the picklist value set from a global picklist. This type extends the Metadata metadata type and inherits its `fullName` field.

Note: GlobalPicklistValue components don't have file suffixes or directories because they're lists of values and not custom fields. For file-based operations they can be accessed through GlobalPicklist (which is in API v37.0 only).

Version

GlobalPicklistValue components are available in API version 37.0 only. In API version 38.0 and later, GlobalPicklistValue is replaced by CustomValue on page 409.

Fields

Field Name	Field Type	Description
`color`	string	The color assigned to the picklist value when it's used in charts on reports and dashboards. The color is in hexadecimal format; for example, #FF6600. If a color is not specified, it's assigned dynamically upon chart generation.
`default`	boolean	Required. Indicates whether this value is the default selection for the global picklist and the custom picklists that share its picklist value set. This field is set to *`true`* by default.
`description`	string	The global picklist value's description. It's useful to include a description for a global picklist value so the reason for creating it can be tracked. Limit: 255 characters.
`isActive`	boolean	Indicates whether this value is currently active or inactive. The default value is *`true`*. Users can select only active values from a picklist. An API retrieve operation for global picklist values returns all active and inactive values in the picklist. (Meanwhile, retrieving the values of a non-global, unrestricted picklist returns only the active values.)

PicklistValue

This metadata type defines a value in the picklist and specifies whether this value is the default value. This type extends the GlobalPicklistValue metadata type and inherits all its fields. In API version 36.0 and earlier, PicklistValue extends the Metadata type and inherits its `fullName` field.

Note the following when working with picklist values:

- When you retrieve a standard object, all picklist values are retrieved, not just the customized picklist values.
- When you deploy changes to standard picklist fields, picklist values are added as needed.
- To deactivate a global picklist value, you can invoke an `update()` call on GlobalPicklist with the value omitted, or with the value's `isActive` field set to `false`. Or, you can invoke an `update()` call directly on GlobalPicklistValue with the `isActive` field set to `false`.

- If picklist values are missing from a component definition, they get deactivated when deployed. Deactivation occurs for picklist values of both standard and custom fields.

Field Name	Field Type	Description
`allowEmail`	boolean	Indicates whether this value lets users email a quote PDF (`true`), or not (`false`). This field is only relevant for the `Status` field in quotes. This field is available in API version 18.0 and later.
`closed`	boolean	Indicates whether this value is associated with a closed status (`true`), or not (`false`). This field is only relevant for the standard `Status` field in cases and tasks. This field is available in API version 16.0 and up to version 36.0. In version 37.0, this field is in GlobalPicklistValue.
`controllingFieldValues`	string[]	A list of values in the controlling field that are linked to this picklist value. The controlling field can be a checkbox or a picklist. This field is available in API version 14.0 and later. The values in the list depend on the field type: • `Checkbox`: `checked` or `unchecked`. • `Picklist`: The `fullname` of the picklist value in the controlling field.
`converted`	boolean	Indicates whether this value is associated with a converted status (`true`), or not (`false`). This field is relevant for only the standard `Lead Status` field in leads. Your organization can set its own guidelines for determining when a lead is qualified, but typically, you want to convert a lead as soon as it becomes a real opportunity that you want to forecast. For more information, see "Convert Qualified Leads" in the Salesforce online help. This field is available in API version 16.0 and later.
`cssExposed`	boolean	Indicates whether this value is available in your Self-Service Portal (`true`), or not (`false`). This field is only relevant for the standard `Case Reason` field in cases. Self-Service provides an online support channel for your customers - allowing them to resolve their inquiries without contacting a customer service representative. For more information about Self-Service, see "Setting Up Your Self-Service Portal" in the Salesforce online help. Note: Starting with Spring '12, the Self-Service portal isn't available for new orgs. Existing orgs continue to have access to the Self-Service portal. This field is available in API version 16.0 and later.
`forecastCategory`	ForecastCategories (enumeration of type string)	Indicates whether this value is associated with a forecast category (`true`), or not (`false`). This field is only relevant for the standard `Stage` field in opportunities. • Omitted • Pipeline • BestCase

Field Name	Field Type	Description
		- Forecast - Closed This field is available in API version 16.0 and later.
highPriority	boolean	Indicates whether this value is a high priority item (true), or not (false). This field is only relevant for the standard Priority field in tasks. For more information about tasks, see "Start Using Tasks" in the Salesforce online help. This field is available in API version 16.0 and later.
probability	int	Indicates whether this value is a probability percentage (true), or not (false). This field is only relevant for the standard Stage field in opportunities. This field is available in API version 16.0 and later.
reverseRole	string	A picklist value corresponding to a reverse role name for a partner. If the role is "subcontractor", then the reverse role might be "general contractor". Assigning a partner role to an account in Salesforce creates a reverse partner relationship so that both accounts list the other as a partner. This field is only relevant for partner roles. For more information, see "Partner Fields" in the Salesforce online help. This field is available in API version 18.0 and later.
reviewed	boolean	Indicates whether this value is associated with a reviewed status (true), or not (false). This field is only relevant for the standard Status field in solutions. For more information about opportunities, see "Creating Solutions" in the Salesforce online help. This field is available in API version 16.0 and later.
won	boolean	Indicates whether this value is associated with a closed or won status (true), or not (false). This field is only relevant for the standard Stage field in opportunities. This field is available in API version 16.0 and later.

Declarative Metadata Sample Definition

For an example of GlobalPicklistValue components with a package.xml that references them, see GlobalPicklist.

Wildcard Support in the Manifest File

This metadata type doesn't support the wildcard character * (asterisk) in the package.xml manifest file. For information about using the manifest file, see Deploying and Retrieving Metadata with the Zip File.

GlobalValueSet

Represents the metadata for a global picklist value set, which is the set of shared values that custom picklist fields can use. A global value set isn't a field itself. (In contrast, the custom picklist fields that are based on a global picklist are of type ValueSet.) This type extends the Metadata metadata type and inherits its fullName field.

File Suffix and Directory Location

GlobalValueSet components have the suffix `.globalValueSet` and are stored in the `globalValueSets` folder.

Version

GlobalValueSet components are available in API version 38.0 and later. In API version 37.0, this is the GlobalPicklist type.

Fields

Field Name	Field Type	Description
`customValue`	CustomValue[]	Requires at least one value. The list of values (or “global value set”) that’s defined for a global picklist. The global value set is inherited by any custom picklist field that uses that value set. Each value is of type customValue. A global value set can have up to 1,000 total values (inclusive of inactive values).
`description`	string	It’s useful to state the global value set’s purpose, and which objects it’s intended for. Limit: 255 characters.
`masterLabel`	string	Required. A global value set’s name, which is defined when the global value set is created. Appears as Label in the user interface.
`sorted`	boolean	Required. Indicates whether a global value set is sorted in alphabetical order. By default this value is `false`.

Declarative Metadata Sample Definition

The following UpsellGlobal.globalValueSet is an example of a GlobalValueSet component.

```
<?xml version="1.0" encoding="UTF-8"?>
<GlobalValueSet xmlns="http://soap.sforce.com/2006/04/metadata">
   <description>Updated:This is a basic global value set.</description>
   <masterLabel>UpsellGlobal</masterLabel>
   <customValue>
      <fullName>Maybe</fullName>
      <default>false</default>
      <label>Maybe</label>
   </customValue>
   <customValue>
      <fullName>No</fullName>
      <default>false</default>
      <label>No</label>
   </customValue>
   <customValue>
      <fullName>Yes</fullName>
      <default>false</default>
      <label>Yes</label>
   </customValue>
```

```
    <sorted>false</sorted>
</GlobalValueSet>
```

The following is an example `package.xml` that references the previous definition.

```
<?xml version="1.0" encoding="UTF-8"?>
<Package xmlns="http://soap.sforce.com/2006/04/metadata">
    <types>
        <members>UpsellGlobal</members>
        <name>GlobalValueSet</name>
    </types>
    <version>40.0</version>
</Package>
```

Wildcard Support in the Manifest File

This metadata type supports the wildcard character `*` (asterisk) in the `package.xml` manifest file. For information about using the manifest file, see Deploying and Retrieving Metadata with the Zip File.

GlobalValueSetTranslation

Contains details for a global value set translation. Global value sets are lists of values that can be shared by multiple custom picklist fields, optionally across objects.This type extends the Metadata metadata type and inherits its `fullName` field.

File Suffix and Directory Location

GlobalValueSetTranslation components have the suffix `.globalValueSetTranslation` and are stored in the `globalValueSetTranslations` folder.

Translations are stored in a file with a format of `ValueSetName-lang.globalValueSetTranslation`, where *`ValueSetName`* is the global value set's name, and lang is the translation language.

Version

GlobalValueSetTranslation components are available in API version 38.0 and later.

Fields

Field	Field Type	Description
`valueTranslation`	ValueTranslation[]	The translated name of a value in a translated global value set. Each `valueTranslation` is paired with a `masterLabel`, which is the original (untranslated) name of the value.

ValueTranslation

The original value name and the translated value name in a translated global value set.

Field	Field Type	Description
`masterLabel`	string	Required. The original (untranslated) name of a value in a global value set. Each `valueTranslation` has a `masterLabel` paired with its `translation`..
`translation`	string	The translated name of a value in a translated global value set.

Declarative Metadata Sample Definition

The following is an example of a GlobalValueSetTranslation component. When a value isn't translated, its translation becomes a comment that's paired with its masterLabel.

```
<?xml version="1.0" encoding="UTF-8"?>
<GlobalValueSetTranslation xmlns="http://soap.sforce.com/2006/04/metadata">
    <valueTranslation>
        <masterLabel>Three</masterLabel>
        <translation>Trois</translation>
    </valueTranslation>
    <valueTranslation>
        <masterLabel>Four</masterLabel>
        <translation>Quatre</translation>
    </valueTranslation>
    <valueTranslation>
        <masterLabel>Five</masterLabel>
        <translation><!-- Five --></translation>
    </valueTranslation>
</GlobalValueSetTranslation>
```

The following is an example `package.xml` that references the previous definition.

```
<?xml version="1.0" encoding="UTF-8"?>
<Package xmlns="http://soap.sforce.com/2006/04/metadata">
        <types>
        <members>Numbers-fr</members>
        <name>GlobalValueSetTranslation</name>
    </types>
    <version>38.0</version>
</Package>
```

Wildcard Support in the Manifest File

This metadata type supports the wildcard character `*` (asterisk) in the `package.xml` manifest file. For information about using the manifest file, see Deploying and Retrieving Metadata with the Zip File.

SEE ALSO:

Translations

Group

Represents a set of public groups, which can have users, roles, and other groups.

Declarative Metadata File Suffix and Directory Location

The file suffix for group components is `.group` and components are stored in the `groups` directory of the corresponding package directory.

Version

Group components are available in API version 24.0 and later.

Special Access Rules

As of Spring ’20 and later, Customer Portal users and guest users can’t access this object.

Fields

Note: Members of the public group are not migrated when you deploy the group type.

This metadata type represents the valid values that define a group:

Field Name	Field Type	Description
`doesIncludeBosses`	boolean	Indicates whether the managers have access (`true`) or do not have access (`false`) to records shared with members of the group. This field is only available for public groups.
`fullName`	string	The unique identifier for API access. The `fullName` can contain only underscores and alphanumeric characters. It must be unique, begin with a letter, not include spaces, not end with an underscore, and not contain two consecutive underscores. This field is inherited from the Metadata component. Corresponds to **Group Name** in the user interface.
`name`	string	Required. The name of the group. Corresponds to **Label** in the user interface.

Declarative Metadata Sample Definition

The following is the definition of a group.

```
<?xml version="1.0" encoding="UTF-8"?>
<Group xmlns="http://soap.sforce.com/2006/04/metadata">
    <doesIncludeBosses>true</doesIncludeBosses>
    <fullName>admin</fullName>
    <name>test</name>
</Group>
```

Wildcard Support in the Manifest File

This metadata type supports the wildcard character * (asterisk) in the package.xml manifest file. For information about using the manifest file, see Deploying and Retrieving Metadata with the Zip File.

HomePageComponent

Represents the metadata associated with a home page component. You can customize the Home tab in Salesforce Classic to include components such as sidebar links, a company logo, a dashboard snapshot, or custom components that you create. Use to create, update, or delete home page component definitions.

For more information, see "Customize Salesforce Classic Home Tab Page Layouts" in the Salesforce online help. This type extends the Metadata metadata type and inherits its fullName field.

Declarative Metadata File Suffix and Directory Location

The file suffix for home page components is .homePageComponent and components are stored in the homepagecomponents directory of the corresponding package directory.

Version

Home page components are available in API version 12.0 and later.

HomePageComponent

This metadata type represents the valid values that define a home page component:

Field Name	Field Type	Description
body	string	If this is an HTML page component, this is the body of the HTML.
fullName	string	The name can only contain characters, letters, and the underscore (_) character, must start with a letter, and cannot end with an underscore or contain two consecutive underscore characters. Inherited from the Metadata component, this field is not defined in the WSDL for this component. It must be specified when creating, updating, or deleting. See create() to see an example of this field specified for a call.
height	int	Required for Visualforce Area components. Indicates the height (in pixels) of the component. This field is available in API version 31.0 and later.
links	string[]	If the pageComponentType is links, then zero or more names of custom page links can be specified. • ObjectWebLink • CustomPageWebLink

Field Name	Field Type	Description
`page`	string	This field is only available for Visualforce Area components and indicates the API name of the Visualforce page that is associated with the component. This field is available in API version 31.0 and later.
`pageComponentType`	PageComponentType (enumeration of type string)	Required. Valid values are: • `links` • `htmlArea` • `imageOrNote` • `visualforcePage` (This value is available in API version 31.0 and later.)
`showLabel`	boolean	This field is only available for Visualforce Area components and specifies whether the component displays with a label (`true`) or not (`false`). This field is available in API version 31.0 and later.
`showScrollbars`	boolean	This field is only available for Visualforce Area components and specifies whether the component displays with scrollbars (`true`) or not (`false`). This field is available in API version 31.0 and later.
`width`	PageComponentWidth (enumeration of type string)	This field is only available for HTML and Visualforce Area components, and indicates whether this is a narrow or wide home page component. Valid values are: • `narrowComponents` • `wideComponents`

Declarative Metadata Sample Definition

The following is the definition of a home page component. See HomePageLayout and WebLink for related samples.

```
<?xml version="1.0" encoding="UTF-8"?>
<HomePageComponent xmlns="http://soap.sforce.com/2006/04/metadata">
  <height>200</height>
  <page>MyVisualforcePage</page>
  <pageComponentType>visualforcePage</pageComponentType>
  <showLabel>true</showLabel>
  <showScrollbars>true</showScrollbars>
  <width>wideComponents</width>
</HomePageComponent>
```

Wildcard Support in the Manifest File

This metadata type supports the wildcard character `*` (asterisk) in the `package.xml` manifest file. For information about using the manifest file, see Deploying and Retrieving Metadata with the Zip File.

SEE ALSO:

HomePageLayout

WebLink

HomePageLayout

Represents the metadata associated with a home page layout. You can customize home page layouts and assign the layouts to users based on their user profile.

For more information, see "Customize Salesforce Classic Home Tab Page Layouts" in the Salesforce online help.

File Suffix and Directory Location

Home page layouts are stored in the `homePageLayouts` directory of the corresponding package directory. The extension is `.homePageLayout`.

Version

Home page components are available in API version 12.0 and later. This type extends the Metadata metadata type and inherits its `fullName` field.

Fields

This metadata type represents the valid values that define a home page layout:

Field Name	Field Type	Description
`fullName`	string	The name can only contain characters, letters, and the underscore (_) character, must start with a letter, and cannot end with an underscore or contain two consecutive underscore characters. Inherited from the Metadata component, this field is not defined in the WSDL for this component. It must be specified when creating, updating, or deleting. See `create()` to see an example of this field specified for a call.
`narrowComponents`	string[]	The list of elements in the narrow column on the left side of the home page.
`wideComponents`	string[]	The list of elements in the wide column on the right side of the home page.

Declarative Metadata Sample Definition

The following is the definition of a home page layout. See HomePageComponent on page 592 and WebLink on page 380 for related samples.

```
<?xml version="1.0" encoding="UTF-8"?>
<HomePageLayout xmlns="http://soap.sforce.com/2006/04/metadata">
    <narrowComponents>google</narrowComponents>
</HomePageLayout>
```

Wildcard Support in the Manifest File

This metadata type supports the wildcard character `*` (asterisk) in the `package.xml` manifest file. For information about using the manifest file, see Deploying and Retrieving Metadata with the Zip File.

SEE ALSO:

- HomePageComponent
- WebLink

InstalledPackage

Represents a package to be installed or uninstalled. Deploying a newer version of a currently installed package upgrades the package.

Note: You can't deploy a package along with other metadata types. When you deploy InstalledPackage, it must be the only metadata type specified in the manifest file.

File Suffix and Directory Location

The package is specified in the `installedPackages` directory, in a file named after the package's namespace prefix. The file extension is `.installedPackage`.

Version

InstalledPackage is available in API version 28.0 and later.

Fields

Field Type	Description	
`activateRSS`	boolean	Required. Determines the state of Remote Site Settings (RSS) and Content Security Policy (CSP) at the time of installing the package and must be set to either of these values. **true** Keep the isActive state of any RSS or CSP in the package.

Field Type	Description	
		false Override the isActive state of any RSS or CSP in the package and set it to `false`. The default value is `false`. Available in API version 43.0 and later.
`password`	string	Specifies the package password.
`versionNumber`	string	Required. The version number of the package. The version number has the format ***`majorNumber.minorNumber.patchNumber`*** (for example, `2.1.3`).

Declarative Metadata Sample Definition

The following example specifies a sample package to be installed or uninstalled.

```
<?xml version="1.0" encoding="UTF-8"?>
        <InstalledPackage xmlns="http://soap.sforce.com/2006/04/metadata">
        <versionNumber>1.0</versionNumber>
        <password>optional_password</password>
        <activateRSS>true</activateRSS>
        </InstalledPackage>
```

Wildcard Support in the Manifest File

This metadata type supports the wildcard character `*` (asterisk) in the `package.xml` manifest file. For information about using the manifest file, see Deploying and Retrieving Metadata with the Zip File.

KeywordList

Represents a list of keywords used in community moderation. This keyword list is a type of moderation criteria that defines offensive language or inappropriate content that you don't want in your community.

This type extends the Metadata metadata type and inherits its `fullName` field.

Keep the following things in mind when creating keyword list criteria:

- Your org can have up to 30 keyword list criteria. This limit is per org, not per community.
- A keyword list can have up to 2,000 keywords.
- Capitalization and trailing punctuation are ignored when matching your keywords to user-generated content. For example, if your criteria includes *`BadWord`*, it's matched when a user types *`BADWORD`* or *`badword.`*

File Suffix and Directory Location

KeywordList components have the suffix `.keywords` and are stored in the `moderation` directory of the corresponding package directory. The file name format follows *`community_name.keyword_list_developer_name`*`.keywords`.

Version

KeywordList components are available in API version 36.0 and later.

Special Access Rules

To view, create, edit, and delete a keyword list, you need the Manage Communities or Create and Set Up Communities permission. As of Spring '20 and later, only users with permission to edit moderation rules can access this object.

Fields

Field Name	Field Type	Description
Description	string	A description of the keyword list.
keywords	Keyword[]	The keywords you want moderate in your community.
masterLabel	string	Required. Label for the keyword list.

Keyword

Keywords in the keyword list.

Field Name	Field Type	Description
keyword	string	Required. Keywords you want to moderate. • Keywords can only be up to 100 characters and can include letters, numbers, spaces, and special characters. • Wildcard characters aren't supported.

Declarative Metadata Sample Definition

The following is an example of a KeywordList component.

```
<?xml version="1.0" encoding="UTF-8"?>
<KeywordList xmlns="http://soap.sforce.com/2006/04/metadata">
   <masterLabel>Bad Word List</masterLabel>
   <description>List of bad words updated by Joe in Nov 2015.</description>
   <keywords>
     <keyword>bad-word</keyword>
   </keywords>
   <keywords>
     <keyword>b a d w o r d</keyword>
   </keywords>
   <keywords>
     <keyword>b@dword</keyword>
   </keywords>
</KeywordList>
```

The following is an example `package.xml` that references the previous definition.

```
<?xml version="1.0" encoding="UTF-8"?>
<Package xmlns="http://soap.sforce.com/2006/04/metadata">
   <types>
     <name>KeywordList</name>
     <members>community1.badword_list</members>
   </types>
   <version>36.0</version>
</Package>
```

Wildcard Support in the Manifest File

This metadata type supports the wildcard character `*` (asterisk) in the `package.xml` manifest file. For information about using the manifest file, see Deploying and Retrieving Metadata with the Zip File.

Layout

Represents the metadata associated with a page layout. For more information, see "Page Layouts" in the Salesforce Help.

This type extends the Metadata metadata type and inherits its `fullName` field.

Note: To edit the Ideas layout, specify it by name in the `package.xml` file. In `package.xml`, use the following code to retrieve the Ideas layout:

```
<types>
    <members>Idea-Idea Layout</members>
    <name>Layout</name>
</types>
```

File Suffix and Directory Location

Layouts are stored in the `layouts` directory of the corresponding package directory. The extension is `.layout`.

Note: Retrieving a component of this metadata type in a project makes the component appear in any Profile and PermissionSet components that are retrieved in the same package.

Version

Layouts are available in API version 13.0 and later.

Fields

This metadata type represents the valid values that define a page layout.

Field Name	Field Type	Description
`customButtons`	string[]	The custom buttons for this layout. Each button is a reference to a WebLink on the same object. For example,

Field Name	Field Type	Description
		a ButtonLink refers to a Weblink on the same standard or custom object named 'ButtonLink'.
`customConsoleComponents`	CustomConsoleComponents	Represents custom console components (Visualforce pages, lookup fields, or related lists; Canvas apps not available) on a page layout. Custom console components only display in the Salesforce console.
`emailDefault`	boolean	Only relevant if `showEmailCheckbox` is set; indicates the default value of that checkbox.
`excludeButtons`	string[]	List of standard buttons to exclude from this layout. For example, `<excludeButtons>Delete</excludeButtons>` excludes the **Delete** button from this layout.
`feedLayout`	FeedLayout	Represents the values that define the feed view of a feed-based page layout. Feed-based layouts are available on Account, Case, Contact, Lead, Opportunity, custom, and external objects. They include a feed view and a detail view.
`headers`	LayoutHeader[] (enumeration of type string)	Layout headers are currently only used for tagging, and only appear in the UI if tagging is enabled. Valid string values are: • `PersonalTagging`—tag is private to user. • `PublicTagging`—tag is viewable any other user who can access the record.
`layoutSections`	LayoutSection[]	The main sections of the layout containing fields, s-controls, and custom links. The order here determines the layout order.
`miniLayout`	MiniLayout	A mini layout is used in the mini view of a record in the Console tab, hover details, and event overlays.
`multilineLayoutFields`	string[]	Fields for the special multiline layout fields which appear in OpportunityProduct layouts. These fields are otherwise similar to `miniLayoutFields` `miniLayout`.
`platformActionList`	PlatformActionList	The list of actions, and their order, that display in the Salesforce app action bar for the layout. This field is available in API version 34.0 and later.
`quickActionList`	QuickActionList	The list of quick actions that display in the full Salesforce site for the page layout. This field is available in API version 28.0 and later.
`relatedContent`	RelatedContent	The Related Content section of the page layout. This field is available in API version 29.0 and later.

Field Name	Field Type	Description
`relatedLists`	RelatedListItem[]	The related lists for the layout, listed in the order they appear in the user interface.
`relatedObjects`	string[]	The list of related objects that appears in the mini view of the console. In database terms, these objects are foreign key fields on the object for the layout. For more information, see “Choose Related Objects for the Agent Console's Mini View” in the Salesforce Help.
`runAssignmentRulesDefault`	boolean	Only relevant if `showRunAssignmentRulesCheckbox` is set; indicates the default value of that checkbox.
`showEmailCheckbox`	boolean	Only allowed on Case, CaseClose, and Task layouts. If set, a checkbox appears to show email.
`showHighlightsPanel`	boolean	If set, the highlights panel displays on pages in the Salesforce console. This field is available in API version 22.0 and later.
`showInteractionLogPanel`	boolean	If set, the interaction log displays on pages in the Salesforce console. This field is available in API version 22.0 and later.
`showKnowledgeComponent`	boolean	Only allowed on Case layouts. If set, the Knowledge sidebar displays on cases in the Salesforce console. This field is available in API version 20.0 and later.
`showRunAssignmentRulesCheckbox`	boolean	Only allowed on Lead and Case objects. If set, a checkbox appears on the page to show assignment rules.
`showSolutionSection`	boolean	Only allowed on CaseClose layout. If set, the built-in solution information section shows up on the page.
`showSubmitAndAttachButton`	boolean	Only allowed on Case layout. If set, the **Submit & Add Attachment** button displays on case edit pages to portal users in the Customer Portal.
`summaryLayout`	SummaryLayout	Controls the appearance of the highlights panel in Salesforce Classic, which summarizes key fields in a grid at the top of a page layout, when Case Feed is enabled. This field is available in API version 18.0 and later.

CustomConsoleComponents

Represents custom console components (Visualforce pages, lookup fields, or related lists; Canvas apps not available) on a page layout. Custom console components only display in the Salesforce console. Available in API version 25.0 and later.

Field Name	Field Type	Description
`primaryTabComponents`	PrimaryTabComponents	Represents custom console components on primary tabs in the Salesforce console. Available in API version 25.0 and later.

Field Name	Field Type	Description
subtabComponents	SubtabComponents	Represents custom console components on subtabs in the Salesforce console. Available in API version 25.0 and later.

PrimaryTabComponents

Represents custom console components on primary tabs in the Salesforce console. Available in API version 25.0 and later.

Field Name	Field Type	Description
component	ConsoleComponent[]	Represents a custom console component (Visualforce page, lookup field, or related lists; Canvas apps not available) on a section of a page layout. Custom console components only display in the Salesforce console. This field is available in API version 29.0 and earlier.
containers	Container[]	Represents a location and style in which to display more than one custom console component on the sidebars of the Salesforce console. You can specify up to five components for each of the four locations (left, right, top, and bottom). This field is available in API version 30.0 and later.

ConsoleComponent

Represents a custom console component (Visualforce page, lookup field, or related lists; Canvas apps not available) on a section of a page layout. Custom console components only display in the Salesforce console. Available in API version 25.0 and later.

Field Name	Field Type	Description
height	int	Required for components with a location of top or bottom. The height of the custom console component. The value must be specified in pixels and be greater than 0 but less than 999.
location	string	Required. The location of the custom console component on the page layout. Valid values are right, left, top, and bottom. A component can have one location for each page layout.
visualforcePage	string	Required. The unique name of the custom console component. For example, ConsoleComponentPage.
width	int	Required for components with a location of left or right. The width of the custom console component. The value must be specified in pixels and be greater than 0 but less than 999.

Container

Represents a location and style in which to display more than one custom console component in the sidebars of the Salesforce console. For example, you can display multiple components in the right sidebar of the console with a style of either stack, tabs, or accordion. Available in API version 30.0 and later.

Field Name	Field Type	Description
`height`	int	Required for components with a location of top or bottom. The height of the components' container. The `unit` field determines the unit of measurement, in pixels or percent.
`isContainerAutoSizeEnabled`	boolean	Required. If set to `true`, stacked console components in the sidebars autosize vertically. Set to `true` by default for newly created console components. Available in API version 32.0 and later.
`region`	string	Required. The location of the components' container. Valid values include: • `right` • `left` • `top` • `bottom`
`sidebarComponents`	SidebarComponent[]	Represents a specific custom console component to display in the components' container.
`style`	string	Required. The style of the container in which to display multiple components. Valid values include: • `stack`—a content area with multiple frames. • `tabs`—a single content area with a list of multiple panels. • `accordian`—a collapsible content area.
`unit`	string	Required. The unit of measurement, in pixels or percent, for the height or width of the components' container. Pixel values are simply the number of pixels, for example, `500`, and must be greater than 0 but less than 999. Percentage values must include the percent sign, for example, `20%`, and must be greater than 0 but less than 100.
`width`	int	Required for components with a location of right or left. The width of the components' container. The `unit` field determines the unit of measurement, in pixels or percent.

SidebarComponent

Represents a specific custom console component to display in a container that hosts multiple components in one of the sidebars of the Salesforce console. You can specify up to five components for each of the four container locations (left, right, top, and bottom). Available in API version 30.0 and later.

Field Name	Field Type	Description
`componentType`	string	Specifies the component type. Valid values are `KnowledgeOne`, `Lookup`, `Milestones`, `RelatedList`, `Topics`, `Files`, and `CaseExperts`. This field is available in API version 31.0 and later. The

Field Name	Field Type	Description
		Files and CaseExperts values are available in API version 32.0 and later. Note: Case Experts is available through a pilot program.
createAction	string	If the component is a lookup field, the name of the quick action used to create a record. This field is available in API version 42.0 and later.
enableLinking	boolean	If the component is a lookup field, lets users associate a record with this field. This field is available in API version 42.0 and later.
height	int	Required for components with a location of top or bottom. The height of the component in the container. The unit field determines the unit of measurement, in pixels or percent.
label	string	The name of the component as it displays to console users. Available for components in a container with the style of tabs or accordion.
lookup	string	If the component is a lookup field, the name of the field.
page	string	If the component is a Visualforce page, the name of the Visualforce page.
relatedlists	RelatedList[]	If the component is a related list, the name of the list. This field is available in API version 31.0 and later.
unit	string	The unit of measurement, in pixels or percent, for the height or width of the component in the container. Pixel values are simply the number of pixels, for example, 500, and must be greater than 0 but less than 999. Percentage values must include the percent sign, for example, 20%, and must be greater than 0 but less than 100.
updateAction	string	If the component is a lookup field, the name of the quick action used to update a record. This field is available in API version 42.0 and later.
width	int	Required for components with a location of right or left. The width of the component in the container. The unit field determines the unit of measurement, in pixels or percent.

RelatedList

Represents related list custom components on the sidebars of the Salesforce console. Available in API version 31.0 and later.

Field Name	Field Type	Description
hideOnDetail	boolean	If set to true, the related list is hidden from detail pages where it appears as a component to prevent duplicate information from showing.
name	string	The name of the component as it appears to console users.

SubtabComponents

Represents custom console components on subtabs in the Salesforce console. Available in API version 25.0 and later.

Field Name	Field Type	Description
`component`	ConsoleComponent[]	Represents a custom console component (Visualforce page, lookup field, or related lists; Canvas apps not available) on a section of a page layout. Custom console components only display in the Salesforce console. This field is available in API version 29.0 and earlier.
`containers`	Container[]	Represents a location and style in which to display more than one custom console component on the sidebars of the Salesforce console. You can specify up to five components for each of the four locations (left, right, top, and bottom). This field is available in API version 30.0 and later.

FeedLayout

Represents the values that define the feed view of a feed-based page layout. Feed-based layouts are available on Account, Case, Contact, Lead, Opportunity, custom, and external objects. They include a feed view and a detail view. Available in API version 30.0 and later.

Field Name	Field Type	Description
`autocollapsePublisher`	boolean	Specifies whether the publisher is automatically collapsed when the page loads (`true`) or not (`false`).
`compactFeed`	boolean	Specifies whether the feed-based page layout uses a compact feed (`true`) or not (`false`). If set to `true`, feed items on the page are collapsed by default, and the feed view has an updated design.
`feedFilterPosition`	FeedLayoutFilterPosition (enumeration of type string)	Where the feed filters list is included in the layout. Valid values are: • `centerDropDown`—as a dropdown list in the center column. • `leftFixed`—as a fixed list in the left column. • `leftFloat`—as a floating list in the left column.
`feedFilters`	FeedLayoutFilter[]	The individual filters displayed in the feed filters list.
`fullWidthFeed`	boolean	Specifies whether the feed expands horizontally to take up all available space on the page (`true`) or not (`false`).
`hideSidebar`	boolean	Specifies whether the sidebar is hidden (`true`) or not (`false`).
`leftComponents`	FeedLayoutComponent[]	The individual components displayed in the left column of the feed view.
`rightComponents`	FeedLayoutComponent[]	The individual components displayed in the right column of the feed view.

FeedLayoutComponent

Represents a component in the feed view of a feed-based page layout. Available in API version 30.0 and later.

Field Name	Field Type	Description
`componentType`	FeedLayoutComponentType (enumeration of type string)	Required. The type of component. Valid values are: • `HelpAndToolLinks`—icons that link to the help topic for the page, the page layout, and, the printable view of the page. Available only on Case layouts. • `CustomButtons`—a custom button. • `Following`—an icon that toggles between a Follow button (if the user viewing a record doesn't already follow it) and a Following indicator (if the user viewing a record does follow it). • `Followers`—a list of users who follow the record. • `CustomLinks`—a custom link. • `Milestones`—the milestone tracker, which lets users see the status of a milestone on a case. Available only on Case layouts. • `Topics`—a list of topics related to the record. • `CaseUnifiedFiles`—a list of all files that are attached to the case. • `Visualforce`—a custom Visualforce component.
`height`	int	The height, in pixels, of the component. Doesn't apply to `standardComponents`
`page`	string	The name of a Visualforce page being used as a custom component.

FeedLayoutFilter

Represents a feed filter option in the feed view of a feed-based page layout. A filter must have only `standardFilter` or `feedItemType` set. Available in API version 30.0 and later.

Field Name	Field Type	Description
`feedFilterName`	string	The name of a CustomFeedFilter component. Names are prefixed with the name of the parent object. For example, *`Case.MyCustomFeedFilter`*.
`feedFilterType`	FeedLayoutFilterType (enumeration of type string)	The type of filter. Valid values are: • `AllUpdates`—shows all feed items on a record. • `FeedItemType`—shows feed items only for a particular type of activity on the record.
`feedItemType`	FeedItemType (enumeration of type string)	The type of feed item to display. Valid values are: • `ActivityEvent`—feed items related to activity on tasks and events associated with a case. Available only on Case layouts. • `AdvancedTextPost`–feed items related to group announcements posted on a feed. This value is available in API version 31.0 and later.

Field Name	Field Type	Description
		• `AnnouncementPost`–Not used. • `ApprovalPost`—feed items related to approvals that are submitted on a feed. • `AttachArticleEvent`—feed items for activity related to attaching articles to cases. Available only on Case layouts. • `BasicTemplateFeedItem`—Not used. • `CallLogPost`—feed items for activity from the Log a Call action. Available only on layouts for objects that support Activities (tasks and events). • `CanvasPost`—feed items related to posts that a canvas app makes on a feed. • `CaseCommentPost`—feed items for activity from the Case Note action. Available only on Case layouts. • `ChangeStatusPost`—feed items for activity from the Change Status action. Available only on Case layouts. • `ChatTranscriptPost`—feed items for activity related to attaching Chat transcripts to cases. Available only on Case layouts. • `CollaborationGroupCreated`—feed items related to creating a public group. • `CollaborationGroupUnarchived`—Not used. • `ContentPost`—feed items related to attaching a file to a post. • `CreatedRecordEvent`—feed items related to creating a record from the publisher. • `DashboardComponentSnapshot`—feed items related to posting a dashboard snapshot on a feed. • `EmailMessageEvent`—feed items for activity from the Email action. Available only on Case layouts. • `FacebookPost`—Not used. • `LinkPost`—feed items related to attaching a URL to a post. • `MilestoneEvent`—feed items for changes to the milestone status on a case. Available only on Case layouts. • `PollPost`—feed items related to posting a poll on a feed. • `ProfileSkillPost`—feed items related to skills added to a user's Chatter profile. This value is available in API version 31.0 and later. • `QuestionPost`—feed items related to posting a question on a feed. This value is available in API version 31.0 and later. • `ReplyPost`—feed items for activity from the Portal action. Available only on Case layouts. • `RypplePost`—feed items related to creating a Thanks badge in WDC.

Field Name	Field Type	Description
		• SocialPost—feed items for activity on Twitter from the Social Post action. • TextPost—feed items for creating a text post from the publisher. • TrackedChange—feed items related to a change or group of changes to a tracked field. • UserStatus—Not used.

MiniLayout

Represents a mini view of a record in the Console tab, hover details, and event overlays.

Field Name	Field Type	Description
`fields`	string[]	The fields for the mini-layout, listed in the order they appear in the UI. Fields that appear here must appear in the main layout.
`relatedLists`	RelatedListItem[]	The mini related list, listed in the order they appear in the UI. You cannot set sorting on mini related lists. Fields that appear here must appear in the main layout.

LayoutSection

LayoutSection represents a section of a page layout, such as the Custom Links section.

Field Name	Field Type	Description
`customLabel`	boolean	Indicates if this section's label is custom or standard (built-in). Custom labels can be any text, but must be translated. Standard labels have a predefined set of valid values, for example 'System Information', which are automatically translated.
`detailHeading`	boolean	Controls if this section appears in the detail page. In the UI, this setting corresponds to the checkbox in the section details dialog.
`editHeading`	boolean	Controls if this section appears in the edit page.
`label`	string	The label; either standard or custom, based on the `customLabel` flag.
`layoutColumns`	LayoutColumn[]	The columns of the layout, depending on the style. 1, 2, or 3 columns, ordered left to right, are possible.
`style`	LayoutSectionStyle (enumeration of type string)	The style of the layout: • `TwoColumnsTopToBottom` - Two columns, tab goes top to bottom • `TwoColumnsLeftToRight` - Two columns, tab goes left to right

Field Name	Field Type	Description
		- `OneColumn` - One column - `CustomLinks` - Contains custom links only

LayoutColumn

LayoutColumn represents the items in a column within a layout section.

Field Name	Field Type	Description
`layoutItems`	LayoutItem[]	The individual items within a column (ordered from top to bottom).
`reserved`	string	This field is reserved for Salesforce. The field resolves an issue with some SOAP libraries. Any value entered in the field is ignored.

LayoutItem

LayoutItem represents the valid values that define a layout item. An item must have only one of the following values set: component, customLink, field, s-control, page, analyticsCloudComponent, or reportChartComponent.

Field Name	Field Type	Description
`behavior`	UiBehavior (enumeration of type string)	Determines the field behavior. **Note:** KAVs, attempting to explicitly specify UiBehavior will result in an exception. UiBehavior must not be specified. Valid string values: - `Edit`—The layout field can be edited but is not required - `Required`—The layout field can be edited and is required - `Readonly`—The layout field is read-only
`canvas`	string	Reference to a canvas app. This field is available in API version 31.0 and later.
`component`	string	Reference to a component. Value must be *`sfa:socialCard`*. This field is available in API version 30.0 and later. This is only allowed inside a `RelatedContentItem`. *`sfa:socialCard`* is only supported on page layouts for contacts, accounts, and leads.
`customLink`	string	The `customLink` reference. This is only allowed inside a `CustomLink layoutSection`.
`emptySpace`	boolean	Controls if this layout item is a blank space.

Field Name	Field Type	Description
`field`	string	The field name reference, relative to the layout object, for example `Description` or `MyField__c`.
`height`	int	For s-control and pages only, the height in pixels.
`page`	string	Reference to a Visualforce page.
`analyticsCloudComponent`	AnalyticsCloudComponentLayoutItem	Refers to a Analytics dashboard that you can add to a standard or custom object page. This field is available in API version 34.0 and later.
`reportChartComponent`	ReportChartComponentLayoutItem	Refers to a report chart that you can add to a standard or custom object page.
`scontrol`	string	Reference to an s-control.
`showLabel`	boolean	For s-control and pages only, whether to show the label.
`showScrollbars`	boolean	For s-control and pages only, whether to show scrollbars.
`width`	string	For s-control and pages only, the width in pixels or percent. Pixel values are simply the number of pixels, for example, `500`. Percentage values must include the percent sign, for example, `20%`.

AnalyticsCloudComponentLayoutItem

Represents the settings for a Analytics dashboard on a standard or custom page. Available in API version 34.0 and later.

Field Name	Field Type	Description
`assetType`	string	Required. Specifies the type of Analytics asset to add. The available asset type is `dashboard`.
`devName`	string	Required. Unique development name of the dashboard to add.
`error`	string	Error string; only populated if an error occurred in the underlying dashboard.
`filter`	string	Communicates initial dashboard filters for mapping data fields in the dashboard to the object's fields, so that the dashboard shows only the data that's relevant for the record being viewed.
`height`	int	Specifies the height of the dashboard, in pixels. The default is `400`.
`hideOnError`	boolean	Controls whether users see a dashboard that has an error. When this attribute is set to `true`, if the dashboard has an error, the dashboard doesn't appear on the page. When set to `false`, the dashboard appears but doesn't show any data except the error. An error can happen when a user doesn't have access to Analytics or to the dashboard. The default is `true`.

Field Name	Field Type	Description
showSharing	boolean	If set to true, and the dashboard is shareable, then the dashboard shows the Share icon. Users can click the icon to open the Share dialog and post or download from the dashboard. If set to false, the dashboard doesn't show the Share icon. This field is available in API version 37.0 and later.
showTitle	boolean	If true, includes the dashboard's title above the dashboard. If false, the dashboard appears without a title. The default is true.
width	string	Specifies the width of the dashboard, in pixels or percent. Pixel values are simply the number of pixels, for example, 500. Percentage values must include the percent sign, for example, 20%. The default is 100%.

ReportChartComponentLayoutItem

Represents the settings for a report chart on a standard or custom page.

Field Name	Field Type	Description
cacheData	boolean	Indicates whether to use cached data when displaying the chart. When the attribute is set to true, data is cached for 24 hours. If the attribute is set to false, the report is run every time the page is refreshed. This field is available in API version 29.0 and later.
contextFilterableField	string	Unique development name of the field by which a report chart is filtered to return data relevant to the page. If set, the ID field for the parent object of the page or report type is the chart data filter. The parent object for the report type and the page must match for a chart to return relevant data.
error	string	Error string; only populated if an error occurred in the underlying report. This field is available in API version 31.0 and later.
hideOnError	boolean	Controls whether users see a chart that has an error. When there's an error and this attribute is not set, the chart doesn't show any data except the error. An error can happen for many reasons, such as when a user doesn't have access to fields used by the chart or a chart has been removed from the report. Set the attribute to true to hide the chart from a page on error. This field is available in API version 29.0 and later.
includeContext	boolean	If true, filters the report chart to return data that's relevant to the page.
reportName	string	Unique development name of a report that includes a chart.
showTitle	boolean	If true, applies the title from the report to the chart.

Field Name	Field Type	Description
`size`	ReportChartComponentSize (enumeration of type string)	The chart size is medium when no value is specified. Valid values: • `SMALL` • `MEDIUM` • `LARGE`

PlatformActionList

PlatformActionList represents the list of actions, and their order, that display in the Salesforce app action bar for the layout. Available in API version 34.0 and later.

Field Name	Field Type	Description
`actionListContext`	PlatformActionListContext (enumeration of type string)	Required. The context of the action list. Valid values are: • `ActionDefinition` • `Assistant` • `BannerPhoto` • `Chatter` • `Dockable` • `FeedElement` • `Flexipage` • `Global` • `ListView` • `ListViewDefinition` • `ListViewRecord` • `Lookup` • `MruList` • `MruRow` • `ObjectHomeChart` • `Photo` • `Record` • `RecordEdit` • `RelatedList` • `RelatedListRecord`
`platformActionListItems`	PlatformActionListItem[]	The actions in the PlatformActionList.
`relatedSourceEntity`	string	When the `ActionListContext` is RelatedList or RelatedListRecord, this field represents the API name of the related list to which the action belongs.

PlatformActionListItem

PlatformActionListItem represents an action in the PlatformActionList. Available in API version 34.0 and later.

Field Name	Field Type	Description
`actionName`	string	The API name for the action in the list.
`actionType`	PlatformActionType (enumeration of type string)	The type of action. Valid values are: • `ActionLink`—An indicator on a feed element that targets an API, a web page, or a file, represented by a button in the Salesforce Chatter feed UI. • `CustomButton`—When clicked, opens a URL or a Visualforce page in a window or executes JavaScript. • `InvocableAction` • `ProductivityAction`—Productivity actions are predefined and attached to a limited set of objects. Productivity actions include Send Email, Call, Map, View Website, and Read News. Except for the Call action, you can't edit or delete productivity actions. • `QuickAction`—A global or object-specific action. • `StandardButton`—A predefined Salesforce button such as New, Edit, and Delete.
`sortOrder`	int	The placement of the action in the list.
`subtype`	string	The subtype of the action. For quick actions, the subtype is `QuickActionType`. For custom buttons, the subtype is `WebLinkTypeEnum`. For action links, subtypes are `Api`, `ApiAsync`, `Download`, and `Ui`. Standard buttons and productivity actions have no subtype.

QuickActionList

QuickActionList represents the list of actions associated with the page layout. Available in API version 28.0 and later.

Field Name	Field Type	Description
`quickActionListItems`	QuickActionListItem[]	Array of zero or more QuickActionList objects.

QuickActionListItem

QuickActionListItem represents an action in the QuickActionList. Available in API version 28.0 and later.

Field Name	Field Type	Description
`quickActionName`	string	The API name of the action.

RelatedContent

RelatedContent represents the Mobile Cards section of the page layout. Available in API version 29.0 and later.

Field Name	Field Type	Description
`relatedContentItems`	RelatedContentItem[]	A list of layout items in the Mobile Cards section of the page layout.

RelatedContentItem

RelatedContentItem represents an individual item in the RelatedContentItem list. Available in API version 29.0 and later.

Field Name	Field Type	Description
`layoutItem`	LayoutItem	An individual LayoutItem in the Mobile Cards section.

RelatedListItem

RelatedListItem represents a related list in a page layout.

Field Name	Field Type	Description
`customButtons`	string[]	A list of custom buttons used in the related list. For more information, see “Define Custom Buttons and Links” in the Salesforce Help .
`excludeButtons`	string[]	A list of excluded related-list buttons.
`fields`	string[]	A list of fields displayed in the related list. Retrieval of standard fields on related lists uses aliases instead of field or API names. For example, the `Fax`, `Mobile`, and `Home Phone` fields are retrieved as Phone2, Phone3, and Phone4, respectively.
`relatedList`	string	Required. The name of the related list.
`sortField`	string	The name of the field that is used for sorting.
`sortOrder`	SortOrder (enumeration of type string)	If the `sortField` is set, the `sortOrder` field determines the sort order. • `Asc` - sort in ascending order • `Desc` - sort in descending order

SummaryLayout

Controls the appearance of the highlights panel in Salesforce Classic, which summarizes key fields in a grid at the top of a page layout, when Case Feed is enabled. Available in API version 25.0 and later.

Field Name	Field Type	Description
masterLabel	string	Required. The name of the layout label.
sizeX	int	Required. Number of columns in the highlights pane, from 1 through 4 (inclusive).
sizeY	int	Required. Number of rows in each column, either 1 or 2.
sizeZ	int	Reserved for future use. If provided, the setting is not visible to users.
summaryLayoutItems	SummaryLayoutItem[]	Controls the appearance of an individual field and its column and row position within the highlights panel grid, when Case Feed is enabled. At least one is required.
summaryLayoutStyle	SummaryLayoutStyle (enumeration of type string)	Highlights panel style. Valid string values are: • Default • QuoteTemplate • DefaultQuoteTemplate • CaseInteraction • QuickActionLayoutLeftRight (Available in API version 28.0 and later.) • QuickActionLayoutTopDown (Available in API version 28.0 and later.)

SummaryLayoutItem

Controls the appearance of an individual field and its column and row position within the highlights panel grid, when Case Feed is enabled. You can have two fields per each grid in a highlights panel. Available in API version 25.0 and later.

Field Name	Field Type	Description
customLink	string	If the item is a custom link, this is the customLink reference.
field	string	The field name reference, relative to the page layout. Must be a standard or custom field that also exists on the detail page.
posX	int	Required. The item's column position in the highlights panel grid. Must be within the range of sizeX.
posY	int	Required. The item's row position in the highlights panel grid. Must be within the range of sizeY.
posZ	int	Reserved for future use. If provided, the setting is not visible to users.

Declarative Metadata Sample Definition

The following is the definition of a page layout:

```
<?xml version="1.0" encoding="UTF-8"?>
<Layout xmlns="http://soap.sforce.com/2006/04/metadata">
   <customConsoleComponents>
```

```
 <primaryTabComponents>
    <container>
        <region>left</region>
       <style>Stack</style>
       <unit>Pixel</unit>
       <width>101</width>
       <sidebarComponent>
             <width>60</width>
             <page>simplepage1</page>
             <unit>Percentage</unit>
       </sidebarComponent>
       <sidebarComponent>
             <width>40</width>
             <page>Hello_World</page>
             <unit>Percentage</unit>
       </sidebarComponent>
    </container>
 </primaryTabComponents>
 <subtabComponents>
    <component>
       <location>top</location>
       <visualforcePage>ConsoleComponentPage2</visualforcePage>
       <height>200</height>
    </component>
 </subtabComponents>
</customConsoleComponents>
 <customButtons>ButtonLink</customButtons>
 <layoutSections>
     <editHeading>true</editHeading>
     <label>Information</label>
     <layoutColumns>
         <layoutItems>
             <behavior>Required</behavior>
             <field>Name</field>
         </layoutItems>
         <layoutItems>
             <height>180</height>
             <scontrol>LayoutSControl</scontrol>
             <showLabel>true</showLabel>
             <showScrollbars>true</showScrollbars>
             <width>50%</width>
         </layoutItems>
         <layoutItems>
             <reportChartComponent>
                <contextFilterableField>CUST_ID</contextFilterableField>
                <includeContext>true</includeContext>
                <reportName>Open_Accounts_by_Cases</reportName>
                <showTitle>false</showTitle>
                <size>LARGE</size>
             <reportChartComponent>
         </layoutItems>
     </layoutColumns>
     <layoutColumns>
         <layoutItems>
```

```
                <behavior>Edit</behavior>
                <field>OwnerId</field>
            </layoutItems>
            <layoutItems>
                <behavior>Edit</behavior>
                <field>CurrencyIsoCode</field>
            </layoutItems>
        </layoutColumns>
        <style>TwoColumnsTopToBottom</style>
    </layoutSections>
    <layoutSections>
        <editHeading>true</editHeading>
        <label>System Information</label>
        <layoutColumns>
            <layoutItems>
                <behavior>Readonly</behavior>
                <field>CreatedById</field>
            </layoutItems>
            <layoutItems>
                <behavior>Readonly</behavior>
                <field>Alpha1__c</field>
            </layoutItems>
            <layoutItems>
                <height>200</height>
                <page>mcanvasPage</page>
                <showLabel>true</showLabel>
                <showScrollbars>false</showScrollbars>
                <width>100%</width>
            </layoutItems>
        </layoutColumns>
        <layoutColumns>
            <layoutItems>
                <behavior>Readonly</behavior>
                <field>LastModifiedById</field>
            </layoutItems>
            <layoutItems>
                <behavior>Edit</behavior>
                <field>TextArea__c</field>
            </layoutItems>
        </layoutColumns>
        <style>TwoColumnsTopToBottom</style>
    </layoutSections>
    <layoutSections>
        <customLabel>true</customLabel>
        <detailHeading>true</detailHeading>
        <label>Custom Links</label>
        <layoutColumns>
            <layoutItems>
                <customLink>CustomWebLink</customLink>
            </layoutItems>
        </layoutColumns>
        <style>CustomLinks</style>
    </layoutSections>
     <quickActionList>
```

```
        <quickActionListItems>
            <quickActionName>FeedItem.TextPost</quickActionName>
        </quickActionListItems>
        <quickActionListItems>
            <quickActionName>FeedItem.ContentPost</quickActionName>
        </quickActionListItems>
        <quickActionListItems>
            <quickActionName>FeedItem.LinkPost</quickActionName>
        </quickActionListItems>
        <quickActionListItems>
            <quickActionName>FeedItem.PollPost</quickActionName>
        </quickActionListItems>
    </quickActionList>
    <relatedContent>
        <relatedContentItems>
            <layoutItem>
                <component>sfa:socialPanel</component>
            </layoutItem>
        <relatedContentItems>
    </relatedContent>
    <miniLayoutFields>Name</miniLayoutFields>
    <miniLayoutFields>OwnerId</miniLayoutFields>
    <miniLayoutFields>CurrencyIsoCode</miniLayoutFields>
    <miniLayoutFields>Alpha1__c</miniLayoutFields>
    <miniLayoutFields>TextArea__c</miniLayoutFields>
    <miniRelatedLists>
        <relatedList>RelatedNoteList</relatedList>
    </miniRelatedLists>
    <relatedLists>
        <fields>StepStatus</fields>
        <fields>CreatedDate</fields>
        <fields>OriginalActor</fields>
        <fields>Actor</fields>
        <fields>Comments</fields>
        <fields>Actor.Alias</fields>
        <fields>OriginalActor.Alias</fields>
        <relatedList>RelatedProcessHistoryList</relatedList>
    </relatedLists>
    <relatedLists>
        <relatedList>RelatedNoteList</relatedList>
    </relatedLists>
</Layout>
```

The following is an example of a layout using `<summaryLayout>`:

```
<?xml version="1.0" encoding="UTF-8"?>
<Layout xmlns="http://soap.sforce.com/2006/04/metadata">
    <layoutSections>
        <editHeading>true</editHeading>
        <label>System Information</label>
        <layoutColumns>
            <layoutItems>
                <behavior>Readonly</behavior>
                <field>CreatedById</field>
            </layoutItems>
            <layoutItems>
                <behavior>Required</behavior>
                <field>Name</field>
            </layoutItems>
        </layoutColumns>
        <layoutColumns>
            <layoutItems>
                <behavior>Readonly</behavior>
                <field>LastModifiedById</field>
            </layoutItems>
        </layoutColumns>
        <style>TwoColumnsTopToBottom</style>
    </layoutSections>
    <summaryLayout>
        <masterLabel>Great Name</masterLabel>
        <sizeX>4</sizeX>
        <sizeY>2</sizeY>
        <summaryLayoutItems>
            <posX>0</posX>
            <posY>0</posY>
            <field>Name</field>
        </summaryLayoutItems>
    </summaryLayout>
</Layout>
```

The following is an example of a feed-based layout:

```
<Layout>
...
    <feedLayout>
        <leftComponents>
            <componentType>customLinks</componentType>
        </leftComponents>
        <rightComponents>
            <componentType>follow</componentType>
        </rightComponents>
        <rightComponents>
            <componentType>followers</componentType>
        </rightComponents>
        <rightComponents>
            <componentType>visualforce</componentType>
            <page>accountCustomWidget</page>
            <height>200</height>
        </rightComponents>
        <hideSidebar>true</hideSidebar>
        <feedFilterPosition>centerDropDown</feedFilterPosition>
        <feedFilters>
    <feedFilerType>allUpdates</feedFilerType>
        </feedFilters>
        <feedFilters>
    <feedFilerType>feedItemType</feedFilerType>
    <feedItemType>CallLogPost</feedItemType>
        </feedFilters>
        <feedFilters>
    <feedFilerType>feedItemType</feedFilerType>
```

```
        <feedItemType>TextPost</feedItemType>
            </feedFilters>
        </feedLayout>
...
</Layout>
```

Wildcard Support in the Manifest File

This metadata type supports the wildcard character `*` (asterisk) in the `package.xml` manifest file. For information about using the manifest file, see Deploying and Retrieving Metadata with the Zip File.

Letterhead

Represents formatting options for the letterhead in an email template. A letterhead defines the logo, page color, and text settings for your HTML email templates. Use letterheads to ensure a consistent look and feel in your company's emails.

For more information, see "Create Classic Letterheads for Email Templates" in the Salesforce online help. This type extends the Metadata metadata type and inherits its `fullName` field.

File Suffix and Directory Location

The file suffix for letterheads is `.letter` and components are stored in the `letterhead` directory of the corresponding package directory.

Version

Letterheads are available in API version 12.0 and later.

Fields

With the exception of logo, and horizontal and vertical alignment, all of these fields are required.

Field Name	Field Type	Description
`available`	boolean	Required. Indicates whether this letterhead can be used (`true`) or not (`false`), for example, in an email template.
`backgroundColor`	string	Required. The background color, in hexadecimal, for example `#FF6600`.
`bodyColor`	string	Required. The body color in hexadecimal.
`bottomLine`	LetterheadLine (enumeration of type string)	Required. The style for the bottom line. Valid style values include: • `color`. The color of the line in hexadecimal, as a string value. • `height`. The height of the line, as an int value.

Field Name	Field Type	Description
`description`	string	Text description of how this letterhead differs from other letterheads.
`fullName`	string	The internal name of the letterhead, based on the `name`, but with white spaces and special characters escaped out for validity.
`footer`	LetterheadHeaderFooter	Required. The style for the footer.
`header`	LetterheadHeaderFooter	Required. The style for the header.
`middleLine`	LetterheadLine	Required. The style for the middle border line in your letterhead. Valid style values include: • `color`. The color of the line in hexadecimal, as a string value. • `height`. The height of the line, as an int value.
`name`	string	Required. The name of the letterhead.
`topLine`	LetterheadLine	Required. The style for the top horizontal line below the header. Valid style values include: • `color`. The color of the line in hexadecimal, as a string value. • `height`. The height of the line, as an int value.

LetterheadHeaderFooter

LetterheadHeaderFooter represents the properties of a header or footer.

Field	Field Type	Description
`backgroundColor`	string	Required. The background color of the header or footer in hexadecimal format.
`height`	DashboardComponent[]	Required. The height of the header or footer.
`horizontalAlignment`	LetterheadHorizontalAlignment (enumeration of type string)	The horizontal alignment of the header or footer. Valid values are: • `None` • `Left` • `Center` • `Right`
`logo`	string	The logo which is a reference to a document, for example `MyFolder/MyDocument.gif`.

Field	Field Type	Description
verticalAlignment	LetterheadVerticalAlignment (enumeration of type string)	The vertical alignment of the header or footer. Valid values are: • None • Top • Middle • Bottom

LetterheadLine

LetterheadLine represents the properties of a line.

Field	Field Type	Description
color	string	Required. The color of the line in hexadecimal format.
height	int	Required. The height of the line.

Declarative Metadata Sample Definition

```
<?xml version="1.0" encoding="UTF-8"?>
<Letterhead xmlns="http://soap.sforce.com/2006/04/metadata">
    <available>true</available>
    <backgroundColor>#CCCCCC</backgroundColor>
    <bodyColor>#33FF33</bodyColor>
    <bottomLine>
        <color>#3333FF</color>
        <height>5</height>
    </bottomLine>
    <description>INITIAL</description>
    <footer>
        <backgroundColor>#FFFFFF</backgroundColor>
        <height>100</height>
        <horizontalAlignment>Left</horizontalAlignment>
        <verticalAlignment>Top</verticalAlignment>
    </footer>
    <header>
        <backgroundColor>#FFFFFF</backgroundColor>
        <height>100</height>
        <horizontalAlignment>Left</horizontalAlignment>
        <verticalAlignment>Top</verticalAlignment>
    </header>
    <middleLine>
        <color>#AAAAFF</color>
        <height>5</height>
    </middleLine>
    <name>SimpleLetterheadLabel</name>
    <topLine>
        <color>#FF99FF</color>
        <height>5</height>
    </topLine>
</Letterhead>
```

Wildcard Support in the Manifest File

This metadata type doesn't support the wildcard character `*` (asterisk) in the `package.xml` manifest file. For information about using the manifest file, see Deploying and Retrieving Metadata with the Zip File.

LightningBolt

Represents the definition of a Lightning Bolt Solution, which can include custom apps, flow categories, and Lightning Community templates. This type extends the Metadata metadata type and inherits its `fullName` field.

File Suffix and Directory Location

LightningBolt components have the suffix `.lightningBolt` and are stored in the `lightningBolts` folder.

Version

LightningBolt components are available in API version 43.0 and later.

Special Access Rules

To add Lightning Community templates to a Lightning Bolt Solution, enable Salesforce Communities in your org.

Fields

Field Name	Field Type	Description
`category`	LightningBoltCategory (enumeration of type string)	Required. The primary industry that the Lightning Bolt Solution is aimed at. Valid values are: - `Communications` - `Education` - `FinancialServices` - `GeneralBusiness` - `Government` - `HealthcareLifeSciences` - `HighTech` - `Manufacturing` - `Media` - `Nonprofits` - `ProfessionalServices`

Field Name	Field Type	Description
		- `RealEstate` - `Retail` - `TravelTransportationHospitality`
`lightningBoltFeatures`	LightningBoltFeatures[]	The list of feature descriptions of this Lightning Bolt Solution.
`lightningBoltImages`	LightningBoltImages[]	The list of images of this Lightning Bolt Solution.
`lightningBoltItems`	LightningBoltItems[]	The list of items (custom apps, flow categories, and Lightning Community templates) that comprise this Lightning Bolt Solution.
`masterLabel`	string	Required. The label of the Lightning Bolt Solution, which appears on the solution detail page.
`publisher`	string	Required. The name of the partner organization associated with this Lightning Bolt Solution.
`summary`	string	Required. The summary description of the Lightning Bolt Solution.

LightningBoltFeatures

Represents the list of feature descriptions of a Lightning Bolt Solution.

Field Name	Field Type	Description
`description`	string	A description of the feature of the Lightning Bolt Solution.
`order`	int	Required. An integer specifying the position of this feature relative to others in the list. 1 is the first position, and 4 is the max position.
`title`	string	Required. The title of the feature, which appears on the solution detail page.

LightningBoltImages

Represents the list of images of a Lightning Bolt Solution.

Field Name	Field Type	Description
`image`	string	Required. The developer name of the `ContentAsset` type, which is used as a preview image for this Lightning Bolt Solution.
`order`	int	Required. An integer specifying the position of this image relative to others in the list. 1 is the first position, and 3 is the max position.

LightningBoltItems

Represents the list of items (custom apps, flow categories, and Lightning Community templates) that comprise a Lightning Bolt Solution.

Field Name	Field Type	Description
`name`	string	Required. The name of the item, which appears on the solution detail page.
`type`	string	Required. The type of the item included in the Lightning Bolt Solution. Valid values are: • `CommunityTemplateDefinition` • `CustomApplication` • `FlowCategory`

Declarative Metadata Sample Definition

The following is an example of a LightningBolt component.

```xml
<?xml version="1.0" encoding="UTF-8"?>
<LightningBolt xmlns="http://soap.sforce.com/2006/04/metadata"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
    <category>Sales</category>
    <lightningBoltFeatures>
        <description>bb</description>
        <order>1</order>
        <title>aa</title>
    </lightningBoltFeatures>
    <lightningBoltImages>
        <image>prm1</image>
        <order>1</order>
    </lightningBoltImages>
    <lightningBoltItems>
        <name>PolaConsole</name>
        <type>CustomApplication</type>
    </lightningBoltItems>
    <lightningBoltItems>
        <name>Banking_Service_Console</name>
        <type>CustomApplication</type>
    </lightningBoltItems>
    <lightningBoltItems>
        <name>Banking_Service_Portal</name>
        <type>CommunityTemplateDefinition</type>
    </lightningBoltItems>
    <lightningBoltItems>
        <name>Banking_Sales_Portal</name>
        <type>CommunityTemplateDefinition</type>
    </lightningBoltItems>
    <lightningBoltItems>
        <name>myorgdev__updatebenefits</name>
        <type>FlowCategory</type>
    </lightningBoltItems>
    <masterLabel>BoltTe</masterLabel>
    <publisher>aaaa</publisher>
    <summary>This is a summary.</summary>

</LightningBolt>
```

The following is an example `package.xml` that references the previous definition.

```
<?xml version="1.0" encoding="UTF-8"?>
<Package xmlns="http://soap.sforce.com/2006/04/metadata">
    <types>
        <members>BoltTe</members>
        <name>LightningBolt</name>
    </types>
    <version>43.0</version>
</Package>
```

Wildcard Support in the Manifest File

This metadata type supports the wildcard character `*` (asterisk) in the `package.xml` manifest file. For information about using the manifest file, see Deploying and Retrieving Metadata with the Zip File.

LightningComponentBundle

Represents a Lightning web component bundle. A bundle contains Lightning web component resources.

Special Access Rules

LightningComponentBundle components can be created only in orgs with defined namespaces.

Fields

Field Name	Field Type	Description
`apiVersion`	double	A double value that binds the component to a Salesforce API version.
`capabilities`	Capabilities[]	A list of capabilities. A capability is something that a component can do, as opposed to a target, which defines where you can use a component. Available in API version 48.0 and later.
`description`	string	A description of the Lightning web component.
`isExplicitImport`	boolean	Indicates whether imports between files are done explicitly by the developer (`true`) or implicitly by the framework (`false`).
`isExposed`	boolean	Indicates whether a component is usable in a managed package (`true`) or not (`false`).
`lwcResources`	LwcResources[]	A list of resources inside a bundle.
`masterLabel`	string	The component title that appears in the list view.

Field Name	Field Type	Description
`targetConfigs`	base64Binary	Configurations for each target. Each target is a Lightning page type. For example, this configuration allows a Lightning web component to be used on a Contact record page in Lightning App Builder.

```
<targetConfigs>
    <targetConfig targets="lightning__RecordPage">
        <objects>
            <object>Contact</object>
        </objects>
    </targetConfig>
</targetConfigs>
```

Field Name	Field Type	Description
`targets`	Targets[]	A list of targets where the Lightning web component is supported. Each target is a Lightning page type that can be configured in Lightning App Builder.

Capabilities

Represents a list of capabilities. A capability is something that a component can do, as opposed to a target, which defines where you can use a component. Available in API version 48.0 and later.

Field	Field Type	Description
`capability`	string	Specifies something that a component can do. The only valid value is `lightningCommunity__RelaxedCSP`, which enables a component installed from a managed package to run in a community that has Lightning Locker disabled.

LwcResources

Represents a list of resources inside a LightningComponentBundle.

Field	Field Type	Description
`lwcResource`	LwcResource	A resource inside a LightningComponentBundle.

LwcResource

Represents a resource inside a LightningComponentBundle.

Field	Field Type	Description
`filePath`	string	Required. The file path of a resource.
`source`	base64Binary	Required. The content of a resource.

Targets

Represents a list of supported containers for a Lightning web component.

Field	Field Type	Description
`target`	string	Specifies the type of Lightning page the component can be added to in Lightning App Builder. Valid values are: • `lightning__AppPage`—Enables a component to be used on a Lightning app page. • `lightning__HomePage`—Enables a component to be used on a custom Lightning Home page. • `lightning__RecordPage`—Enables a component to be used on a Lightning record page, such as Account or Contact.

Declarative Metadata Sample Definition

This `package.xml` file retrieves all the LightningComponentBundle components in an org.

```
<?xml version="1.0" encoding="UTF-8"?>
<Package xmlns="http://soap.sforce.com/2006/04/metadata">
    <types>
        <members>*</members>
        <name>LightningComponentBundle</name>
    </types>
    <version>45.0</version>
</Package>
```

In the retrieved `zip` file, each Lightning web component is nested under an `lwc` folder.

This example shows the directory structure in the zip file of one component with a name of `hello`.

```
lwc
    hello
        hello.html
        hello.js
        hello.js-meta.xml
```

Here are the contents of the files in the `hello` directory.

Content of `hello.html`:

```
<template>

    <lightning-card title="Hello" icon-name="custom:custom14">
        <div class="slds-m-around_medium">
            Hello, {greeting}!
        </div>
    </lightning-card>
```

```
</template>
```

Content of `hello.js`:

```
import { LightningElement, track } from 'lwc';

export default class Hello extends LightningElement {
    @track greeting = 'World';
}
```

Content of `hello.js-meta.xml`.

```
<?xml version="1.0" encoding="UTF-8"?>
<LightningComponentBundle xmlns="http://soap.sforce.com/2006/04/metadata">
    <apiVersion>45.0</apiVersion>
    <isExposed>true</isExposed>
    <targets>
        <target>lightning__AppPage</target>
        <target>lightning__RecordPage</target>
        <target>lightning__HomePage</target>
    </targets>
</LightningComponentBundle>
```

Wildcard Support in the Manifest File

This metadata type supports the wildcard character `*` (asterisk) in the `package.xml` manifest file. For information about using the manifest file, see Deploying and Retrieving Metadata with the Zip File.

LightningExperienceTheme

Represents the details of a custom theme, including the BrandingSet. Themes enable admins to specify configurable attributes, such as three colors and five images. The colors and some of the images override SLDS token values and influence the generation of `app.css`.

File Suffix and Directory Location

LightningExperienceTheme components have the suffix `.lightningExperienceTheme` and are stored in the `lightningExperienceThemes` folder.

Version

LightningExperienceTheme components are available in API version 42.0 and later.

Special Access Rules

The LightningExperieceTheme type is available when the S1DesktopAllowed permission is enabled in your org.

Fields

Field Name	Field Type	Description
defaultBrandingSet	string	Required. The ID of the BrandingSet properties associated with this LightningExperienceTheme.
description	string	The optional description text of this LightningExperienceTheme. Limited to 1000 characters.
masterLabel	string	Required. The label for this LightningExperienceTheme, which displays in Setup. Limited to 70 characters.
shouldOverrideLoadingImage	boolean	If true, the LightningExperienceTheme overrides the splash screen image.

Declarative Metadata Sample Definition

The following is an example of a LightningExperienceTheme component. See BrandingSet on page 226 for an example of the BrandingSet component.

```
<?xml version="1.0" encoding="UTF-8"?>
<LightningExperienceTheme xmlns="http://soap.sforce.com/2006/04/metadata">
    <defaultBrandingSet>SummerCelebrationBrand</defaultBrandingSet>
    <description>Theme for summer celebration week.</description>
    <masterLabel>Summer Celebration</masterLabel>
    <shouldOverrideLoadingImage>false</shouldOverrideLoadingImage>
</LightningExperienceTheme>
```

The following is an example package.xml that references the previous definition.

```
<?xml version="1.0" encoding="UTF-8"?>
<Package xmlns="http://soap.sforce.com/2006/04/metadata">
    <types>
        <members>LEXTHEMINGThemeName</members>
        <name>BrandingSet</name>
    </types>
    <types>
     <members>Summer Celebration</members>
     <name>LightningExperienceTheme</name>
    </types>
    <version>42.0</version>
</Package>
```

Wildcard Support in the Manifest File

This metadata type supports the wildcard character * (asterisk) in the package.xml manifest file. For information about using the manifest file, see Deploying and Retrieving Metadata with the Zip File.

LightningMessageChannel (Beta)

Represents the metadata associated with a Lightning Message Channel. A Lightning Message Channel represents a secure channel to communicate across UI technologies (Lightning Web Components, Aura Components, and Visualforce). This type extends the Metadata on page 652 metadata type and inherits its `fullName` field.

Note: As a beta feature, LightningMessageChannel is a preview and isn't part of the "Services" under your master subscription agreement with Salesforce. Use this feature at your sole discretion, and make your purchase decisions only on the basis of generally available products and features. Salesforce doesn't guarantee general availability of this feature within any particular time frame or at all, and we can discontinue it at any time. LightningMessageChannel is for evaluation purposes only, not for production use. It's offered as is and isn't supported, and Salesforce has no liability for any harm or damage arising out of or in connection with it. All restrictions, Salesforce reservation of rights, obligations concerning the Services, and terms for related Non-Salesforce Applications and Content apply equally to your use of this feature. You can provide feedback and suggestions in the Trailblazer Community.

File Suffix and Directory Location

LightningMessageChannel components have the suffix `.messageChannel` and are stored in the `messageChannels` folder.

Version

LightningMessageChannel components are available in API version 47.0 and later.

Fields

Field Name	Field Type	Description
`description`	string	The description of the Lightning Message Channel.
`isExposed`	boolean	Indicates whether a Lightning Message Channel is exposed to components in other namespaces (`true`) or not (`false`). The default value is `false`. If you set `isExposed` to `true` on a Message Channel that is in a managed package or referenced by another Message Channel component, you cannot change it to `false`. Other orgs or components already rely on your Message Channel, which can cause their code to break.
`lightningMessageFields`	LightningMessageField on page 629[]	A list of message payload fields for a given Lightning Message Channel.
`masterLabel`	string	Required. The label for a Lightning Message Channel.

LightningMessageField

Represents a message payload field for a given Lightning Message Channel.

Field Name	Field Type	Description
`description`	string	The description for a Lightning Message Field.

Field Name	Field Type	Description
`fieldName`	string	Required. Unique identifier of the Lightning Message Field.

Declarative Metadata Sample Definition

Here's a simple example of a LightningMessageChannel component.

```
<?xml version="1.0" encoding="UTF-8"?>
<LightningMessageChannel xmlns="http://soap.sforce.com/2006/04/metadata">
    <masterLabel>SampleMessageChannel</masterLabel>
    <isExposed>true</isExposed>
    <description>This is a sample Lightning Message Channel.</description>
</LightningMessageChannel>
```

The following is an example of a LightningMessageChannel component with LightningMessageFields.

```
<?xml version="1.0" encoding="UTF-8"?>
<LightningMessageChannel xmlns="http://soap.sforce.com/2006/04/metadata">
    <masterLabel>SampleMessageChannel</masterLabel>
    <isExposed>true</isExposed>
    <description>This is a sample Lightning Message Channel.</description>
    <lightningMessageFields>
        <fieldName>recordId</fieldName>
        <description>This is the record Id that changed</description>
    </lightningMessageFields>
    <lightningMessageFields>
        <fieldName>recordData</fieldName>
        <description>The current data representing the record that changed</description>
    </lightningMessageFields>
</LightningMessageChannel>
```

The following is an example `package.xml` that references the previous definition.

```
<?xml version="1.0" encoding="UTF-8"?>
<Package xmlns="http://soap.sforce.com/2006/04/metadata">
    <types>
        <members>*</members>
        <name>LightningMessageChannel</name>
    </types>
    <version>47.0</version>
</Package>
```

Wildcard Support in the Manifest File

This metadata type supports the wildcard character `*` (asterisk) in the `package.xml` manifest file. For information about using the manifest file, see Deploying and Retrieving Metadata with the Zip File.

LiveChatAgentConfig

Represents the configuration of an organization's Chat deployment, such as how many chats can be assigned to an agent and whether or not chat sounds are enabled.

This type extends the Metadata metadata type and inherits its `fullName` field.

File Suffix and Directory Location

LiveChatAgentConfig configurations are referenced in the `<developer_name>.liveChatAgentConfig` file in the `liveChatAgentConfigs` directory.

Version

LiveChatAgentConfig is available in API version 28.0 and later.

Fields

Field Name	Field Type	Description
`assignments`	AgentConfigAssignments	Specifies how agent configurations are assigned to Chat users. Agent configurations can be assigned to sets of users or sets of profiles.
`autoGreeting`	string	Specifies the greeting that displays when a customer begins a chat with an agent.
`capacity`	int	Specifies the maximum number of chats in which an agent can be engaged at a time.
`criticalWaitTime`	int	Specifies the number of seconds an agent can wait to answer an engaged chat before the chat tab flashes to alert the agent to answer it.
`customAgentName`	string	Specifies the custom name for an agent, if one has been set. Available in API version 29.0 and later.
`enableAgentFileTransfer`	boolean	Indicates whether file transfer is enabled for agents (`true`) or not (`false`). Available in API version 31.0 and later.
`enableAgentSneakPeek`	boolean	Specifies whether a supervisor can see the content of an agent’s message before they send it to a customer (`true`) or not (`false`).
`enableAssistanceFlag`	boolean	Indicates whether agents can raise an assistance flag to notify a supervisor that they need help. Available in API version 35.0 and later.
`enableAutoAwayOnDecline`	boolean	Indicates whether an agent appears as “away” (`true`) or not (`false`) when an agent declines a chat with a customer.
`enableAutoAwayOnPushTimeout`	boolean	Indicates whether an agent appears as “away” (`true`) or not (`false`) when a chat request that's been pushed to the agent times out. Available in API version 34.0 and later.

Field Name	Field Type	Description
enableChatConferencing	boolean	Indicates whether chat conferencing is enabled for agents (true) or not (false). Available in API version 34.0 and later.
enableChatMonitoring	boolean	Indicates whether chat monitoring is enabled for support supervisors (true) or not (false). Available in API version 29.0 and later.
enableChatTransferToAgent	boolean	Indicates whether agents can transfer a chat to another agent (true) or not (false). Available in API version 36.0 and later.
enableChatTransferToButton	boolean	Indicates whether agents can transfer a chat to a button (true) or not (false). Available in API version 36.0 and later.
enableChatTransferToSkill	boolean	Indicates whether agents can transfer a chat to a skill group (true) or not (false). Available in API version 36.0 and later.
enableLogoutSound	boolean	Indicates whether a sound will play (true) or not (false) when an agent logs out of Chat.
enableNotifications	boolean	Indicates whether notifications of incoming chats appear for agents (true) or not (false).
enableRequestSound	boolean	Indicates whether a sound will play (true) or not (false) when a customer requests to chat with an agent.
enableSneakPeek	boolean	Indicates whether previews of customers' messages are displayed as customers type (true) or not (false) in the agent's Chat window. Available in API version 29.0 and later.
enableVisitorBlocking	boolean	Indicates whether an agent can block a visitor by IP address (true) or not (false). Available in API version 34.0 and later.
enableWhisperMessage	boolean	Indicates whether support supervisors can send whisper messages to agents during a chat (true) or not (false). Available in API version 29.0 and later.
label	string	Required. Specifies the name of the configuration for agents' default chat settings.
supervisorDefaultAgentStatusFilter	SupervisorAgentStatusFilter (enumeration of type string)	Specifies the Chat status for filtering the Agent Status list in the Supervisor Panel. Valid values are: • Online • Away • Offline Available in API version 29.0 and later.

Field Name	Field Type	Description
supervisorDefaultButtonFilter	string	Specifies the default button for filtering the Agent Status list in the Supervisor Panel. Available in API version 29.0 and later.
supervisorDefaultSkillFilter	string	Specifies the default skill for filtering the Agent Status list in the Supervisor Panel. Available in API version 29.0 and later.
supervisorSkills	SupervisorAgentConfigSkills	Specifies the list of agent skills that are assigned to a supervisor, as specified in their assigned Chat configuration. Available in API version 29.0 and later.
transferableButtons	AgentConfigButtons	Specifies the list of chat buttons that agents can transfer chats to. Available in API version 31.0 and later.
transferableSkills	AgentConfigSkills	Specifies the list of skill groups that agents can transfer chats to. Available in API version 31.0 and later.

AgentConfigAssignments

Represents the assignments of an organization’s profiles and users to a Chat configuration.

Field Name	Field Type	Description
profiles	AgentConfigProfileAssignments	Specifies the profiles that are associated with a specific agent configuration.
users	AgentConfigUserAssignments	Specifies the users that are associated with a specific agent configuration.

AgentConfigButtons

Represents the chat buttons that agents who are associated with the Chat configuration can transfer chats to.

Field Name	Field Type	Description
button	string[]	Specifies the chat buttons that agents can transfer chats to.

AgentConfigProfileAssignments

Represents the profiles associated with a specific Chat configuration.

Field Name	Field Type	Description
profile	string	Specifies the custom name of the profile associated with a specific agent configuration.

AgentConfigSkills

Represents the skill groups that agents who are associated with the Chat configuration can transfer chats to.

Field Name	Field Type	Description
`skill`	string[]	Specifies the skill groups that agents can transfer chats to.

AgentConfigUserAssignments

Represents the users associated with a specific Chat configuration.

Field Name	Field Type	Description
`user`	string	Specifies the username of the user associated with a specific agent configuration.

SupervisorAgentConfigSkills

Represents the agent skills associated with a supervisor's Chat configuration. Available in API version 29.0 and later.

Field Name	Field Type	Description
`skill`	string	Specifies the agent skills available for filtering the Agent Status list in the Supervisor Panel.

Declarative Metadata Sample Definition

This is a sample of a `liveChatAgentConfig` file.

```
<?xml version="1.0" encoding="UTF-8"?>
<LiveChatAgentConfig xmlns="http://soap.sforce.com/2006/04/metadata">
    <label>My Agent Configuration 1</label>
    <autoGreeting>Hi, how can I help you?</autoGreeting>
    <capacity>5</capacity>
    <enableAutoAwayOnDecline>true</enableAutoAwayOnDecline>
    <enableLogoutSound>true</enableLogoutSound>
    <enableNotifications>true</enableNotifications>
    <enableRequestSound>true</enableRequestSound>
    <enableSneakPeek>true</enableSneakPeek>
    <enableWhisperMessage>true</enableWhisperMessage>
    <assignments>
        <profiles>
            <profile>standard</profile>
        </profiles>
        <users>
            <user>jdoe@acme.com</user>
        </users>
    </assignments>
</LiveChatAgentConfig>
```

Wildcard Support in the Manifest File

This metadata type supports the wildcard character `*` (asterisk) in the `package.xml` manifest file. For information about using the manifest file, see Deploying and Retrieving Metadata with the Zip File.

LiveChatButton

Represents a Chat deployment's settings for the button that customers click to chat with an agent and the chat window, such as the label that appears on the button and the pre-chat form that appears before a chat begins.

This type extends the Metadata metadata type and inherits its `fullName` field.

Note: Chats routed with Omni-Channel aren't supported in the Metadata API.

File Suffix and Directory Location

LiveChatButton configurations are stored in the `<developer_name>.liveChatButton` file in the `liveChatButtons` directory.

Version

LiveChatButton is available in API version 28.0 and later.

Fields

Field Name	Field Type	Description
`animation`	LiveChatButtonPresentation (enumeration of type string)	The type of animation for a chat invitation. Valid values are: • `Slide` • `Fade` • `Appear` • `Custom`
`autoGreeting`	string	The customized greeting message that the customer receives when an agent accepts a chat request from the chat button or invitation. Available in API version 29.0 and later.
`chasitorIdleTimeout`	int	Specifies the amount of idle time before the chat times out. The idle time starts being counted after the agent sends the last chat message. Available in API version 35.0 and later.
`chasitorIdleTimeoutWarning`	int	Specifies the amount of idle time before a warning appears. The idle time starts being

Field Name	Field Type	Description
		counted after the agent sends the last chat message. Available in API version 35.0 and later.
`chatPage`	string	Specifies the page that hosts your chat if that page differs from the Chat window.
`customAgentName`	string	The agent's name as it appears to customers in the chat window. Available in API version 29.0 and later.
`deployments`	LiveChatButtonDeployments	Specifies the deployments associated with the button.
`enableQueue`	boolean	Indicates whether queuing is enabled (`true`) or not (`false`).
`inviteEndPosition`	LiveChatButtonInviteEndPosition (enumeration of type string)	The end position of the chat invitation. Valid values include: • `TopLeft` • `Top` • `TopRight` • `Left` • `Center` • `Right` • `BottomLeft` • `Bottom` • `BottomRight`
`inviteImage`	string	The custom button graphic that appears for the invitation.
`inviteStartPosition`	LiveChatButtonInviteStartPosition (enumeration of type string)	The start position of the chat invitation. Valid values include: • `TopLeft` • `TopLeftTop` • `Top` • `TopRightTop` • `TopRight` • `TopRightRight` • `Right` • `BottomRightRight` • `BottomRight` • `BottomRightBottom` • `Bottom`

Field Name	Field Type	Description
		- BottomLeftBottom - BottomLeft - BottomLeftLeft - Left - TopLeftLeft
isActive	boolean	Specifies whether the chat button or invitation is active.
label	string	Specifies the text that appears on the button.
numberOfReroutingAttempts	int	Specifies the number of times a chat request can be rerouted to available agents if all agents reject the chat request. Available in API version 30.0 and later.
offlineImage	string	Specifies the image that appears on the button when no agents are available to chat.
onlineImage	string	Specifies the image that appears on the button when agents are available to chat.
optionsCustomRoutingIsEnabled	boolean	Indicates whether custom routing is enabled for incoming chat requests (true) or not (false). Available in API version 30.0 and later.
optionsHasChasitorIdleTimeout	boolean	Indicates whether the visitor idle timeout feature is enabled. Available in API version 35.0 and later.
optionsHasInviteAfterAccept	boolean	Indicates whether a new chat invitation triggers after a customer accepts a previous chat invitation (true) or not (false).
optionsHasInviteAfterReject	boolean	Indicates whether a new chat invitation triggers after a customer rejects a previous chat invitation (true) or not (false).
optionsHasRerouteDeclinedRequest	boolean	Indicates whether a chat request, which has been rejected by all available agents, should be rerouted to available agents again (true) or not (false). Available in API version 30.0 and later.
optionsIsAutoAccept	boolean	Indicates whether a chat request should be automatically accepted by the agent it's assigned to (true) or not (false). For chat buttons and automated chat invitations with routingType set to MostAvailable or LeastActive. Available in API version 30.0 and later.

Field Name	Field Type	Description
`optionsIsInviteAutoRemove`	boolean	Indicates whether a chat invitation is set to automatically disappear from a customer's screen after a certain amount of time (`true`) or not (`false`).
`overallQueueLength`	int	Specifies the maximum number of chat requests that are allowed to queue.
`perAgentQueueLength`	int	Specifies the number of chat requests that are allowed to queue for an agent with the required skills.
`postChatPage`	string	Specifies the name of the post-chat form to which customers are routed when the chat ends.
`postChatUrl`	string	Specifies the URL of the post-chat form to which customers are routed when the chat ends.
`preChatFormPage`	string	Specifies the name of the pre-chat form to which customers are routed before a chat begins.
`preChatFormUrl`	string	Specifies the URL of the pre-chat form to which customers are routed when the chat begins.
`pushTimeOut`	int	Specifies the number of seconds an agent has to answer an incoming chat request before the request is routed to another agent.
`routingType`	LiveChatButtonRoutingType (enumeration of type string)	Specifies how incoming chats should be routed to agents when a customer pushes a button. Valid values are: • `Choice` • `LeastActive` • `MostAvailable`
`site`	string	Specifies the Salesforce site that hosts your custom chat button images or custom chat page. Note: You must have the "CustomDomain" permission enabled in your organization before you can use a Salesforce site with Chat.
`skills`	LiveChatButtonSkills	Specifies the skills associated with the button. When a customer clicks the button to chat, they are automatically routed to agents with those skills.
`timeToRemoveInvite`	int	Specifies how long the invitation is displayed (in seconds) to customers before it disappears.

Field Name	Field Type	Description
`type`	LiveChatButtonType (enumeration of type string)	Required. The chat button type. Valid values are: • `Standard` • `Invite`
`windowLanguage`	Language	Specifies the language preferences for the chat window associated with the button.

LiveChatButtonSkills

Represents the skills associated with a chat button or invitation.

Fields

Field Name	Field Type	Description
`skill`	string	Specifies the name of the skill.

LiveChatButtonDeployments

Represents the deployments associated with a chat button or invitation.

Fields

Field Name	Field Type	Description
`deployment`	string	Specifies the name of the deployment.

Declarative Metadata Sample Definition

This is a sample of a `liveChatButton` file.

```
<?xml version="1.0" encoding="UTF-8"?>
<LiveChatButton xmlns="http://soap.sforce.com/2006/04/metadata">
    <deployments/>
    <enableQueue>false</enableQueue>
    <isActive>true</isActive>
    <label>CustomerSupportButton</label>
    <optionsCustomRoutingIsEnabled>false</optionsCustomRoutingIsEnabled>
    <optionsHasChasitorIdleTimeout>false</optionsHasChasitorIdleTimeout>
    <optionsHasInviteAfterAccept>false</optionsHasInviteAfterAccept>
    <optionsHasInviteAfterReject>false</optionsHasInviteAfterReject>
    <optionsHasRerouteDeclinedRequest>false</optionsHasRerouteDeclinedRequest>
    <optionsIsAutoAccept>false</optionsIsAutoAccept>
    <optionsIsInviteAutoRemove>false</optionsIsInviteAutoRemove>
    <postChatUrl>http://help.salesforce.com</postChatUrl>
    <routingType>Choice</routingType>
    <skills>
        <skill>Chat</skill>
    </skills>
    <type>Standard</type>
</LiveChatButton>
```

Note: If you update your chat button through the Metadata API, be sure to update all Web pages that use the same chat button code.

Wildcard Support in the Manifest File

This metadata type supports the wildcard character `*` (asterisk) in the `package.xml` manifest file. For information about using the manifest file, see Deploying and Retrieving Metadata with the Zip File.

LiveChatDeployment

Represents the configuration settings for a specific Chat deployment, such as the branding image for the deployment and whether or not chat transcripts are automatically saved.

This type extends the Metadata metadata type and inherits its `fullName` field.

File Suffix and Directory Location

LiveChatDeployment values are stored in the `<developer_name>.liveChatDeployment` file in the `liveChatDeployments` directory.

Version

LiveChatDeployment is available in API version 28.0 and later.

Fields

Field Name	Field Type	Description
`brandingImage`	string	Specifies the branding image for the deployment.
`connectionTimeoutDuration`	int	Indicates the amount of time before the chat times out, in seconds.
`ConnectionWarningDuration`	int	Indicates the amount of time before a time-out warning is displayed to the agent, in seconds.
`displayQueuePosition`	boolean	(Pilot) Determines whether a customer's queue position is displayed in a standard chat window while the customer waits for an agent to respond to the chat request (`true`) or not

Field Name	Field Type	Description
		(`false`). This field is available as a pilot in API version 32.0. To enable this field, contact Salesforce.
`domainWhiteList`	LiveChatDeploymentDomainWhiteList	Specifies the list of domains that can host the deployment.
`enablePrechatApi`	boolean	Indicates whether or not the pre-chat API is enabled for the deployment (`true`) or not (`false`).
`enableTranscriptSave`	boolean	Indicates whether chat transcripts are automatically saved after a chat ends (`true`) or not (`false`).
`label`	string	Specifies the name of the deployment.
`mobileBrandingImage`	string	Specifies the branding image for the deployment that appears when customers access the deployment on a mobile device.
`site`	string	Specifies the site that hosts the images for the deployment. Note: You must have the "CustomDomain" permission enabled in your organization before you can use a Salesforce site with Chat.
`windowTitle`	string	Specifies the title of the window associated with the deployment.

LiveChatDeploymentDomainWhiteList

Represents a Chat deployment's domain whitelist.

Fields

Field Name	Field Type	Description
`domain`	string	Specifies a domain that can host the deployment.

Declarative Metadata Sample Definition

This is a sample of a `liveChatDeployment` file.

```
<?xml version="1.0" encoding="UTF-8"?>
<LiveChatDeployment xmlns="http://soap.sforce.com/2006/04/metadata">
```

```
    <label>My Deployment 1</label>
    <brandingImage>pkb_image_bannerBg</brandingImage>
    <mobileBrandingImage>pkb_image_bgBottom</mobileBrandingImage>
    <domainWhiteList>
        <domain>mydomain</domain>
        <domain>test</domain>
    </domainWhiteList>
    <enableTranscriptSave>true</enableTranscriptSave>
    <site>GL_Knowledge_Base</site>
    <windowTitle>My window title</windowTitle>
</LiveChatDeployment>
```

Note: If you update your deployment through the Metadata API, be sure to update all Web pages that use the same deployment code.

Wildcard Support in the Manifest File

This metadata type supports the wildcard character `*` (asterisk) in the `package.xml` manifest file. For information about using the manifest file, see Deploying and Retrieving Metadata with the Zip File.

LiveChatSensitiveDataRule

Represents a rule for masking or deleting data of a specified pattern. Written as a regular expression (regex).

Use this object to mask or delete data of specified patterns, such as credit card, social security, phone and account numbers, or even profanity. This type extends the Metadata metadata type and inherits its `fullName` field.

File Suffix and Directory Location

LiveChatSensitiveDataRule components have the suffix `.liveChatSensitiveDataRule` and are stored in the `liveChatSensitiveDataRule` folder.

Version

LiveChatSensitiveDataRule components are available in API version 35.0 and later.

Fields

Field Name	Field Type	Description
`actionType`	SensitiveDataActionType (enumeration of type string)	Required. The action to take on the text when the sensitive data rule is triggered. Possbile values are: - `Remove` - `Replace`
`description`	string	The description of the sensitive data rule—for example, “Block social security numbers.”

Field Name	Field Type	Description
`enforceOn`	int	Required. Determines the roles on which the rule is enforced. The value is determined using bitwise OR operation. There are seven possible values: **1.** Rule enforced on Agent **2.** Rule enforced on Visitor **3.** Rule enforced on Agent and Visitor **4.** Rule enforced on Supervisor **5.** Rule enforced on Agent and Supervisor **6.** Rule enforced on Visitor and Supervisor **7.** Rule enforced on Agent, Visitor, and Supervisor
`isEnabled`	boolean	Required. Specifies whether a sensitive data rule is active (`true`) or not (`false`). Default value (if none is provided) is `false`.
`pattern`	string	Required. The pattern of text blocked by the rule. Written as a JavaScript regular expression (regex).
`replacement`	string	The string of characters that replaces the blocked text (if `ActionType` *`Replace`* is selected).

Declarative Metadata Sample Definition

The following is an example of a LiveChatSensitiveDataRule component.

```
<LiveChatSensitiveDataRule xmlns="http://soap.sforce.com/2006/04/metadata">
    <actionType>REPLACE</actionType>
    <enforceOn>7</enforceOn>
    <isEnabled>true</isEnabled>
    <pattern>[aeiou]</pattern>
    <replacement>œ</replacement>
</LiveChatSensitiveDataRule>
```

The following is an example `package.xml` that references the previous definition.

```
<Package xmlns="http://soap.sforce.com/2006/04/metadata">
    <!-- To be used from
support.liveagent.testsuite.unifiedouting.testDeployButtonMDAPIWithExistingQueue -->
    <apiAccessLevel>Unrestricted</apiAccessLevel>

    <types>
        <members>Change_For_all</members>
        <name>LiveChatSensitiveDataRule</name>
    </types>

    <version>35.0</version>
</Package>
```

Wildcard Support in the Manifest File

This metadata type supports the wildcard character `*` (asterisk) in the `package.xml` manifest file. For information about using the manifest file, see Deploying and Retrieving Metadata with the Zip File.

ManagedContentType

Represents the definition of custom content types for use with Salesforce CMS. Custom content types are displayed as forms with defined fields. This type extends the Metadata metadata type and inherits its `fullName` field.

File Suffix and Directory Location

ManagedContentType components have the suffix `managedContentType` and are stored in the `managedContentTypes` folder.

Version

ManagedContentType components are available in API version 47.0 and later.

Special Access Rules

ManagedContentType is only available if Salesforce CMS and Communities are enabled for your org.

Fields

Field Name	Field Type	Description
`description`	string	Describes the custom content type defined in this ManagedContentType declaration.
`developerName`	string	Required. Unique name for the custom content type. For example: *`OurSpecialContent_c`*
`managedContentNodeTypes`	ManagedContentNodeType on page 644[]	Nodes included as part of this custom content type. When rendered as a form in the Salesforce CMS app, each node is represented as an individual field.
`masterLabel`	string	Required. Declares the name of the content type as it appears in the UI.

ManagedContentNodeType

Represents the structure of individual nodes within the custom content type.

Field Name	Field Type	Description
`helpText`	string	Provides assistive text in the UI, displayed as an infobubble for the field. If this field is empty, no infobubble icon or text is displayed. For example: <IMG?>
`isLocalizable`	boolean	Declares a field as localizable and consumable by <loc MDAPI reference> (`true`) or not (`false`). Default is `false`.
`isRequired`	boolean	Declares a field as required (`true`) or not (`false`). Fields declared as required are indicated by a red asterisk. If a value isn't added to the field in the custom content type, the form can't be saved and a standard error is displayed. Default is `false`. Note: When `nodeType on page 645` is set to `NAMEFIELD` on a field, `isRequired` must also be set to `True` for that field.
`nodeLabel`	string	Required. Declares the label for the field as it appears in the UI.
`nodeName`	string	Required. Unique name of the `nodeType` within the content type.
`nodeType`	MCNodeType (enumeration of type string)	Required. Identifies the supported type of content in the node. Passed as a string. There is a maximum of 15 node types per content type. Values are case insensitive but are returned in all capital letters as shown. Valid values are: • `TEXT` Simple text node (max length=255 characters) • `MTEXT` Multi-line text node (max length=2000 characters) • `RTE` Rich text node (max length=65536 characters) • `IMG` Image node • `URL` URL node (max length=255 characters) Note: `URL` accepts protocol string values starting with http://, https://, mailto:, tel:, and /. • `DATE` Date node Note: `DATE` accepts dates only in the format yyyy-MM-dd. • `DATETIME` Datetime node

Field Name	Field Type	Description
		Note: DATETIME accepts date and time in the format : yyyy-MM-dd'T'HH:mm:ss.SSS'Z' (UTC datetime in ISO 8601 format). • NAMEFIELD Note: NAMEFIELD declares the field as the name that represents the content when referenced in the UI. For example, text entered in this field would be displayed in a list of available content in the Salesforce CMS app or as a piece of content available for inclusion in a collection in a community. One, and only one, nodeType in your managed content type must be declared as NAMEFIELD. – NAMEFIELD is a string of 80 characters or fewer. – When NAMEFIELD is used, isRequired must also be set to true for the field.
placeholderText	string	Provides assistive text in the UI, displayed as placeholder, or ghost text, in a field before any entry is made. For example, Enter a title for your article...

Declarative Metadata Sample Definition

The following is an example of a ManagedContentType component.

```
<?xml version="1.0" encoding="UTF-8"?>
<ManagedContentType xmlns="http://soap.sforce.com/2006/04/metadata">
  <developerName>myContentType</developerName>
 <masterLabel>My Content Type</masterLabel>
 <description>This is the description for my content type</description>
 <managedContentNodeTypes>
  <nodeName>title</nodeName>
  <nodeLabel>Content Title</nodeLabel>
  <nodeType>NAMEFIELD</nodeType>
  <placeholderText>Placeholder Text for title</placeholderText>
  <helpText>Help Text for title</helpText>
  <isLocalizable>true</isLocalizable>
  <isRequired>true</isRequired>
 </managedContentNodeTypes>
 <managedContentNodeTypes>
  <nodeName>textnode</nodeName>
  <nodeLabel>Content Text</nodeLabel>
  <nodeType>TEXT</nodeType>
  <placeholderText>Placeholder Text for Content Text</placeholderText>
  <helpText>Help Text for Content Text</helpText>
  <isLocalizable>true</isLocalizable>
  <isRequired>false</isRequired>
 </managedContentNodeTypes>
 <managedContentNodeTypes>
```

```
  <nodeName>richtextnode</nodeName>
  <nodeLabel>Content RichText</nodeLabel>
  <nodeType>RTE</nodeType>

 </managedContentNodeTypes>
 <managedContentNodeTypes>
  <nodeName>multilinetextnode</nodeName>
  <nodeLabel>Content MultilineText</nodeLabel>
  <nodeType>MTEXT</nodeType>
 </managedContentNodeTypes>
 <managedContentNodeTypes>
  <nodeName>imagenode</nodeName>
  <nodeLabel>Content Image</nodeLabel>
  <nodeType>IMG</nodeType>

 </managedContentNodeTypes>
</ManagedContentType>
```

Usage

For each custom content type you create, there must also be a CMS Content page created in any community that displays the content. Each Content page serves as the detail page for all content of a single content type. See Create Custom Pages with Experience Builder.

Wildcard Support in the Manifest File

This metadata type supports the wildcard character `*` (asterisk) in the `package.xml` manifest file. For information about using the manifest file, see Deploying and Retrieving Metadata with the Zip File.

ManagedTopics

Represents navigational and featured topics managed in a community. A specific community is represented by the Network component.

Note: The related network must exist before you deploy managed topics. (This occurs automatically when deploying an entire organization.)

File Suffix and Directory Location

Components have the suffix managedTopics and are stored in the managedTopics folder. In that folder, you'll find separate files for each network (for example, `NetworkNameA.managedTopics` and `NetworkNameB.managedTopics`).

Version

ManagedTopics components are available in API version 32.0 and later.

Fields

Field Name	Field Type	Description
`ManagedTopic`	ManagedTopic	Represents a specific navigational or featured topic.

ManagedTopic

Field Name	Field Type	Description
`name`	string	The topic name.
`managedTopicType`	string	The topic type: “Navigational” or “Featured”
`topicDescription`	string	An optional description of topic contents. This field is accessible only via the API; there is no corollary in the user interface.
`parentName`	string	The name of a parent topic for which this topic is a child. Child topics are accessible from the subtopics section of the parent topic page and their feeds are added to the parent topic feed. Only navigational topics support parent-child relationships.
`position`	int	The placement of this topic relative to others of the same type. The results differ depending on topic type: • For top-level navigational topics, `position` arranges the Topics menu in the community. • For child navigational topics, it arranges sibling topics in the subtopics section. • For featured topics, it arranges topic thumbnail images on the community home page. Enter a number between 0 and 24. (The maximum amount of navigational or featured topics is 25.)

Declarative Metadata Sample Definition

The following example retrieves or deploys managed topics for all networks:

```
<?xml version="1.0" encoding="UTF-8"?>
<Package xmlns="http://soap.sforce.com/2006/04/metadata">
    <types>
        <members>*</members>
        <name>ManagedTopics</name>
    </types>
    <version>32.0</version>
</Package>
```

The following example shows a package.xml file referencing the ManagedTopics component:

```
<?xml version="1.0" encoding="UTF-8"?>
<Package xmlns="http://soap.sforce.com/2006/04/metadata">
    <types>
        <members>NetworkName</members>
        <name>ManagedTopics</name>
    </types>
    <version>32.0</version>
</Package>
```

The following example shows the ManagedTopics component itself:

```
<?xml version="1.0" encoding="UTF-8"?>
<ManagedTopics>
    <ManagedTopic>
        <name>Running</name>
        <managedTopicType>Navigational</managedTopicType>
        <topicDescription>Training advice</topicDescription>
        <parentName></parentName>
        <position>0</position>
    </ManagedTopic>
    <ManagedTopic>
        <name>Hiking</name>
        <managedTopicType>Navigational</managedTopicType>
        <topicDescription>Routes and gear</topicDescription>
        <parentName></parentName>
        <position>1</position>
    </ManagedTopic>
        <ManagedTopic>
            <name>Trails</name>
            <managedTopicType>Navigational</managedTopicType>
            <topicDescription>Maps for local favorites</topicDescription>
            <parentName>Hiking</parentName>
            <position>0</position>
        </ManagedTopic>
        <ManagedTopic>
            <name>Backpacks</name>
            <managedTopicType>Navigational</managedTopicType>
            <topicDescription>Recommended models</topicDescription>
            <parentName>Hiking</parentName>
            <position>1</position>
        </ManagedTopic>
    <ManagedTopic>
        <name>Footwear</name>
        <managedTopicType>Featured</managedTopicType>
        <topicDescription>Suggested types for each sport</topicDescription>
        <parentName></parentName>
        <position>0</position>
    </ManagedTopic>
    <ManagedTopic>
        <name>Conditioning</name>
        <managedTopicType>Featured</managedTopicType>
        <topicDescription>How to get fit for any activity</topicDescription>
        <parentName></parentName>
        <position>1</position>
    </ManagedTopic>
</ManagedTopics>
```

Wildcard Support in the Manifest File

This metadata type supports the wildcard character `*` (asterisk) in the `package.xml` manifest file. For information about using the manifest file, see Deploying and Retrieving Metadata with the Zip File.

MatchingRule

Represents a matching rule that is used to identify duplicate records. This type extends the Metadata metadata type and inherits its `fullName` field.

File Suffix and Directory Location

Matching rule components have the `.matchingRule` suffix and are stored in the `matchingRules` folder. The name of the component file is the standard or custom object name that is associated with the matching rule.

In API version 39.0 and later, MatchingRule supports the Person Account object.

- The component file name is `PersonAccount.matchingRule`.
- The component directory is `matchingRules`.

Version

MatchingRule is available in API version 33.0 and later.

Fields

Field Name	Field Type	Description
`booleanFilter`	string	Specifies filter logic conditions.
`description`	string	The description of the matching rule.
`label`	string	Required. The name of the matching rule.
`matchingRuleItems`	MatchingRuleItem	The criteria that make up a matching rule.
`ruleStatus`	MatchingRuleStatus (enumeration of type string)	Required. The activation status of the matching rule. Values are: • *`Inactive`* • *`Deactivating`* • *`DeactivationFailed`* • *`Active`* • *`Activating`* • *`ActivationFailed`*

Field Name	Field Type	Description
		Important: The only valid values you can declare when deploying a package are *Active* and *Inactive*.

MatchingRuleItem

Field Name	Field Type	Description
`blankValueBehavior`	BlankValueBehavior (enumeration of type string)	Specifies how blank fields affect whether the fields being compared are considered matches. Valid values are: - *MatchBlanks* - *NullNotAllowed* (default)
`fieldName`	string	Required. Indicates which field to compare when determining if a record is similar enough to an existing record to be considered a match.
`matchingMethod`	MatchingMethod (enumeration of type string)	Required. Defines how the fields are compared. Choose between the exact matching method and various fuzzy matching methods. Valid values are: - *Exact* - *FirstName* - *LastName* - *CompanyName* - *Phone* - *City* - *Street* - *Zip* - *Title* For details on each matching method, see “Matching Methods Used with Matching Rules” in the Salesforce Help.

Declarative Metadata Sample Definition

The following is a sample XML definition of a matching rule. A matching rule can be associated with either a standard or a custom object.

```
<?xml version="1.0" encoding="UTF-8"?>
<MatchingRules xmlns="http://soap.sforce.com/2006/04/metadata">
<matchingRules>
<fullName>AccountMatchingRule</fullName>
<label>Matching rule for accounts</label>
<description>this is sample rule description</description>
<matchingRuleItems>
<blankValueBehavior>NullNotAllowed</blankValueBehavior>
<fieldName>BillingCity</fieldName>
<matchingMethod>City</matchingMethod>
```

```
</matchingRuleItems>

<matchingRuleItems>
<blankValueBehavior>NullNotAllowed</blankValueBehavior>
<fieldName>Name</fieldName>
<matchingMethod>CompanyName</matchingMethod>
</matchingRuleItems>

<ruleStatus>Inactive</ruleStatus>
</matchingRules>
</MatchingRules>
```

The following `package.xml` shows how to reference a matching rule by name. It specifies the type name of MatchingRule.

```
<?xml version="1.0" encoding="UTF-8"?>
<Package xmlns="http://soap.sforce.com/2006/04/metadata">
<types>
<members>Account.AccountMatchingRule</members>
<name>MatchingRule</name>
</types>
<version>48.0</version>
</Package>
```

The following `package.xml` shows how to reference all matching rules by specifying the plural MatchingRules type name and using a wildcard to include all members.

```
<?xml version="1.0" encoding="UTF-8"?>
<Package xmlns="http://soap.sforce.com/2006/04/metadata">
<types>
<members>*</members>
<name>MatchingRules</name>
</types>
<version>48.0</version>
</Package>
```

Wildcard Support in the Manifest File

This metadata type supports the wildcard character `*` (asterisk) in the `package.xml` manifest file. For information about using the manifest file, see Deploying and Retrieving Metadata with the Zip File.

Metadata

This is the base class for all metadata types. You cannot edit this object. A component is an instance of a metadata type.

Metadata is analogous to sObject, which represents all standard objects. Metadata represents all components and fields in Metadata API. Instead of identifying each component with an ID, each custom object or custom field has a unique `fullName`, which must be distinct from standard object names, as it must be when you create custom objects or custom fields in the Salesforce user interface.

Version

Metadata components are available in API version 10.0 and later.

Fields

Field Name	Field Type	Description
`fullName`	string	Required. The name of the component. If a field, the name must specify the parent object, for example `Account.FirstName`. The __c suffix must be appended to custom object names and custom field names when you are setting the `fullName`. For example, a custom field in a custom object could have a `fullName` of `MyCustomObject__c.MyCustomField__c`. To reference a component in a package, prepend the package's namespace prefix to the component name in the `fullName` field. Use the following syntax: ***`namespacePrefix__ComponentName`***. For example, for the custom field component `MyCustomObject__c.MyCustomField__c` and the namespace `MyNS`, the full name is `MyNS__MyCustomObject__c.MyCustomField__c`. Note: A namespace prefix is a 1 to 15-character alphanumeric identifier that distinguishes your package and its contents from other publishers' packages. For more information, see "Register a Namespace Prefix" in the Salesforce Help.

Wildcard Support in the Manifest File

This metadata type doesn't support the wildcard character `*` (asterisk) in the `package.xml` manifest file. For information about using the manifest file, see Deploying and Retrieving Metadata with the Zip File.

SEE ALSO:

- CustomObject
- CustomField
- MetadataWithContent

MetadataWithContent

MetadataWithContent is the base type for all metadata types that contain content, such as documents or email templates. It extends Metadata. You cannot edit this object.

Version

MetadataWithContent components are available in API version 14.0 and later.

Fields

Field Name	Field Type	Description
content	base64Binary	Base 64-encoded binary data. Prior to making an API call, client applications must encode the binary attachment data as base64. Upon receiving a response, client applications must decode the base64 data to binary. This conversion is usually handled for you by a SOAP client.
fullName	string	Required. The name of the component. The fullName can contain only underscores and alphanumeric characters. It must be unique, begin with a letter, not include spaces, not end with an underscore, and not contain two consecutive underscores. Inherited from the Metadata component, this field is not defined in the WSDL for this component. It must be specified when creating, updating, or deleting. See create() to see an example of this field specified for a call.

Wildcard Support in the Manifest File

This metadata type doesn't support the wildcard character * (asterisk) in the package.xml manifest file. For information about using the manifest file, see Deploying and Retrieving Metadata with the Zip File.

SEE ALSO:

Metadata

MilestoneType

Represents the name and description of a milestone, which you can use in an entitlement process to track important steps in cases.

This type extends the Metadata metadata type and inherits its fullName field.

File Suffix and Directory Location

Milestone types are stored in the milestoneTypes directory of the corresponding package directory. The extension is .milestoneType.

Version

MilestoneType is available in API version 27.0 and later.

Fields

Field Name	Field Type	Description
description	string	The description of the milestone.

Field Name	Field Type	Description
RecurrenceType	MilestoneTypeRecurrenceType (enumeration of type string)	The type of recurrence for the milestone. Available in API version 29.0 and later. Valid values are: • none—Specifies no recurrence for the milestone. The milestone occurs only once until the entitlement process exits. • recursIndependently—Specifies independent recurrence for the milestone. • recursChained—Specifies sequential recurrence for the milestone.

Declarative Metadata Sample Definition

This is a sample milestone type.

```
<?xml version="1.0" encoding="UTF-8"?>
<MilestoneType xmlns="http://soap.sforce.com/2006/04/metadata">
    <description>First Response Time</description>
</MilestoneType>
```

And, here's the sample package.xml file that references the MilestoneType component definition:

```
<?xml version="1.0" encoding="UTF-8"?>
<Package xmlns="http://soap.sforce.com/2006/04/metadata">
    <types>
        <members>* or a valid name of a milestone type</members>
        <name>MilestoneType</name>
    </types>
    <version>29.0</version>
</Package>
```

Wildcard Support in the Manifest File

This metadata type supports the wildcard character * (asterisk) in the package.xml manifest file. For information about using the manifest file, see Deploying and Retrieving Metadata with the Zip File.

MlDomain (Beta)

Represents an Einstein Intent Set. This type extends the Metadata metadata type and inherits its fullName field.

NLU features are preview and aren't part of the "Services" under your master subscription agreement with Salesforce. Use this feature at your sole discretion, and make your purchase decisions only on the basis of generally available products and features. Salesforce doesn't guarantee general availability of this feature within any particular time frame or at all, and we can discontinue it at any time. This feature is for evaluation purposes only, not for production use. It's offered as is and isn't supported, and Salesforce has no liability for any harm or damage arising out of or in connection with it. All restrictions, Salesforce reservation of rights, obligations concerning the Services, and terms for related Non-Salesforce Applications and Content apply equally to your use of this feature.

File Suffix and Directory Location

MlDomain components have the suffix `.mlDomain` and are stored in the `mlDomains` folder.

Version

MlDomain components are available in API version 43.0 and later.

Special Access Rules

This object is available only if Chat and Einstein Bots are enabled in your org.

Fields

Field Name	Field Type	Description
`description`	string	Einstein Intent Set description.
`label`	string	Einstein Intent Set name.
`mlIntents`	MlIntent[]	List of intents under this Einstein Intent Set.
`mlSlotClasses`	MlSlotClass[]	List of entities under this Einstein Intent Set.

MlIntent

An intent in an Einstein Intent Set.

Field Name	Field Type	Description
`description`	string	Einstein Intent Set description.
`developerName`	string	Required. This unique name prevents conflicts with other Einstein Intent Sets associated with the same bot version. This name can contain only underscores and alphanumeric characters and must be unique in your org. It must begin with a letter, not include spaces, not end with an underscore, and not contain two consecutive underscores.
`label`	string	Einstein Intent Set name.
`mlIntentUtterances`	MlIntentUtterance[]	List of customer inputs for this intent.
`relatedMlIntents`	MlRelatedIntent[]	List of intents within an Einstein Intent Set used to expand customer inputs for this intent. Only intents within local Einstein Intent Sets have related intents.

MlIntentUtterance

A customer input for this intent.

Field Name	Field Type	Description
`utterance`	string	A customer input or natural language query that triggers the parent intent.

MlRelatedIntent

An intent in an Einstein Intent Set used to expand customer inputs for this intent. Only intents within local Einstein Intent Sets have related intents.

Field Name	Field Type	Description
`relatedMlIntent`	string	Name of the intent that is used to extend the customer inputs of the current parent intent.

MlSlotClass

An entity in this Einstein Intent Set.

Field Name	Field Type	Description
`dataType`	MlSlotClassDataType (enumeration of type string)	A list of the data types available for the MlSlotClass. Valid values are: - `Text` - `Number` - `Boolean` - `Date` - `DateTime` - `Currency`
`description`	string	A description of an Einstein Bot entity.
`developerName`	string	Required. This unique name prevents conflicts with other entities in an Einstein Intent Set. This name can contain only underscores and alphanumeric characters, and must be unique in your org. It must begin with a letter, not include spaces, not end with an underscore, and not contain two consecutive underscores.
`extractionRegex`	string	Regular expression used to extract an entity when the type is set to `Pattern`.
`extractionType`	MlSlotClassExtractionType (enumeration of type string)	Required. Valid values are: - `Pattern` - `Value`
`label`	string	Label that identifies an entity throughout the Salesforce user interface.
`mlSlotClassValues`	MlSlotClassValue[]	List of entity values associated with an entity of type `Value`.

MlSlotClassValue

An entity value associated with an entity of type `Value`.

Field Name	Field Type	Description
`synonymGroup`	SynonymGroup	Represents a list of terms or synonyms for the current entity value.
`value`	string	Single value used to extract an entity of type `Value`.

SynonymGroup

Represents a group of synonymous words or phrases.

Field Name	Field Type	Description
`languages`	Language (enumeration of type string)	Required. Specifies the languages the value list applies to. If value list items are specific to a single language, specify only that language. If the value list items apply to multiple languages, specify multiple languages for one value list.
`terms`	string	Required. A word or phrase synonymous with other terms in the value list.

Declarative Metadata Sample Definition

The following is an example of an MlDomain.

```
<?xml version="1.0" encoding="UTF-8"?>
<MlDomain xmlns="http://soap.sforce.com/2006/04/metadata">
    <label>TestDomainMetadata</label>
    <description>This is domain 2 for metadata testing</description>
    <mlIntents>
        <developerName>Test_Intent_New</developerName>
        <label>Test Intent New</label>
        <mlIntentUtterances>
            <utterance>Utterance Hello</utterance>
        </mlIntentUtterances>
        <mlIntentUtterances>
            <utterance>Utterance Hi</utterance>
        </mlIntentUtterances>
        <mlIntentUtterances>
            <utterance>Utterance What</utterance>
        </mlIntentUtterances>
    </mlIntents>
    <mlIntents>
        <developerName>Test_Intent_New2</developerName>
        <label>Test Intent New 2</label>
    </mlIntents>
    <mlSlotClasses>
        <developerName>Test_Entity1</developerName>
        <label>Test Entity 1</label>
        <extractionType>Value</extractionType>
        <mlSlotClassValues>
```

```
                <value>Choice value 1</value>
            </mlSlotClassValues>
            <mlSlotClassValues>
                <value>Choice value 2</value>
            </mlSlotClassValues>
    </mlSlotClasses>
    <mlSlotClasses>
        <developerName>Test_Entity2</developerName>
        <label>Test Entity 2</label>
        <extractionType>Pattern</extractionType>
    </mlSlotClasses>
    <mlSlotClasses>
            <dataType>Text</dataType>
            <description>Valid Email Address</description>
            <developerName>Email</developerName>
            <extractionRegex>\b[A-Z0-9._%+-]+@[A-Z0-9.-]+\.[A-Z]{2,}\b</extractionRegex>
            <extractionType>Pattern</extractionType>
            <label>Email</label>
    </mlSlotClasses>
    <mlSlotClasses>
        <developerName>airport</developerName>
        <extractionType>Value</extractionType>
        <label>airport</label>
        <mlSlotClassValues>
            <synonymGroup>
                <languages>en_US</languages>
                <terms>San Francisco</terms>
                <terms>The City</terms>
            </synonymGroup>
            <value>SFO</value>
        </mlSlotClassValues>
        <mlSlotClassValues>
            <synonymGroup>
                <languages>en_US</languages>
                <terms>Oakland</terms>
                <terms>The Town</terms>
            </synonymGroup>
            <value>OAK</value>
        </mlSlotClassValues>
    </mlSlotClasses>
</MlDomain>
```

The following is an example `package.xml` that references the previous definition.

```
<?xml version="1.0" encoding="UTF-8"?>
<Package xmlns="http://soap.sforce.com/2006/04/metadata">
    <types>
        <members>TestDomainMetadata</members>
        <name>MlDomain</name>
    </types>
    <version>43.0</version>
</Package>
```

Wildcard Support in the Manifest File

This metadata type supports the wildcard character `*` (asterisk) in the `package.xml` manifest file. For information about using the manifest file, see Deploying and Retrieving Metadata with the Zip File.

MobileApplicationDetail

Represents the packaging attributes for a mobile connected app. This type extends the Metadata metadata type and inherits its `fullName` field.

File Suffix and Directory Location

MobileApplicationDetail components have the suffix `MobileApplicationDetail` and are stored in the `MobileApplicationDetails` folder.

Version

MobileApplicationDetail components are available in API version 47.0 and later.

Fields

Field Name	Field Type	Description
`applicationBinaryFile`	base64	Base 64-encoded binary data file for the mobile app.
`applicationBinaryFileName`	string	Filename for the mobile app binary data file.
`applicationBundleIdentifier`	string	iOS apps only: the unique application bundle identifier.
`applicationFileLength`	int	The length of the mobile app binary data file.
`applicationIconFile`	string	iOS apps only: the application icon.
`applicationIconFileName`	string	iOS apps only: the application icon filename.
`applicationInstallUrl`	string	URL to install the mobile app.
`devicePlatform`	DevicePlatformType (enumeration of type string)	Required. Platform that supports the mobile app. The valid values are: • `android` • `ios`
`deviceType`	string	Supported device type for mobile app. The valid values are: • `minitablet` • `phone` • `tablet`
`minimumOsVersion`	string	Minimum OS version required to install the mobile app.
`privateApp`	boolean	Specifies whether the mobile app is private (`true`) or not (`false`).

Field Name	Field Type	Description
`version`	string	Required. Version number of the mobile app.

Usage

When you create a connected app in Salesforce Classic or Lightning Experience and enter mobile app settings, those settings are stored in a MobileApplicationDetail component. In this example, the metadata retrieved for a connected app includes MobileApplicationDetail metadata.

```
<?xml version="1.0" encoding="UTF-8"?>
<<ConnectedApp xmlns="http://soap.sforce.com/2006/04/metadata">
  <contactEmail>paul.chen@salesforce.com</contactEmail>
  <label>MobileApplicationDetailConnectedApp</label>
  <mobileAppConfig>
    <applicationBinaryFile></applicationBinaryFile>
    <applicationInstallUrl>https://appstore.apple.com/MobileApplicationDetail
        </applicationInstallUrl>
    <devicePlatform>ios</devicePlatform>
    <deviceType>phone</deviceType>
    <privateApp>false</privateApp>
    <version>0.0.0.0</version>
  </mobileAppConfig>
< . mobileStartUrl>https://www.salesforce.com</mobileStartUrl>
</ConnectedApp>
```

Wildcard Support in the Manifest File

This metadata type doesn't support the wildcard character `*` (asterisk) in the `package.xml` manifest file. For information about using the manifest file, see Deploying and Retrieving Metadata with the Zip File.

ModerationRule

Represents a rule used in your community to moderate member-generated content.Each rule specifies the member-generated content the rule applies to, the criteria to enforce the rule on, and the moderation action to take. Moderation rules help protect your community from spammers, bots, and offensive or inappropriate content. This type extends the Metadata metadata type and inherits its `fullName` field.

Community moderation rules created with the Metadata API are more powerful than moderation rules set up in the Community Management UI. For example, in the UI you could create a rule that moderates posts and comments. In the Metadata API you could create a rule that moderates only the Link Name of a Link feed type. Use the Metadata API to express complex moderation rules.

Important: Don't update moderation rules you create using the Metadata API in the Community Management UI. If you do, you overwrite relevant Metadata API fields or the fields are ignored.

Keep the following things in mind when creating moderation rules:

- Your org can have up to 30 rules. This limit is per org, not per community. This limit includes both content rules and rate rules.
- Each rule can have up to three keyword criteria.
- Rules that block content run first, followed by rules to review and approve content, then rules that replace content, and last by rules that flag content. If two or more rules perform the same action, the oldest rule runs first, based on the date the rule was created.

Rules to replace content don't run when the content also applies to a review rule—we want community managers to review the original content.

File Suffix and Directory Location

ModerationRule components have the suffix `.rule` and are stored in the `moderation` directory of the corresponding package directory. The file name format follows *`community_name.moderation_rule_developer_name`*`.rule`.

Version

ModerationRule components are available in API version 36.0 and later.

Special Access Rules

To view, create, edit, and delete moderation rules, you need the Manage Communities or Create and Set Up Communities permission. As of Spring '20 and later, only users with permission to edit moderation rules can access this object.

Fields

Field Name	Field Type	Description
`action`	ModerationRuleAction (enumeration of type string)	Required. Indicates the moderation action that you want to take. The valid values are: • `Block` • `Review` • `Replace` • `Flag` • `FreezeAndNotify` (Reserved for future use.)
`actionLimit`	int	Indicates the moderation action limit. Available in API 39.0 and later.
`active`	boolean	Required. Indicates whether the moderation rule is active (`true`) or inactive (`false`).
`description`	string	A description of the moderation rule.
`entitiesAndFields`	ModeratedEntityField[]	Indicates the types of user-generated content this moderation rule applies to.
`masterLabel`	string	Required. Label for the moderation rule.
`notifyLimit`	int	Indicates the notification limit of the moderation rule. Available in API 39.0 and later.
`userCriteria`	string	Represents the member criteria to use in community moderation rules. Available in API 39.0 and later.
`userMessage`	string	The message you want your community members to see when their content is blocked. Use the *`%BLOCKED_KEYWORD%`* variable to display

Field Name	Field Type	Description
		up to five blocked words in the user message. If you don't specify a message, the member sees the standard message: "You can't use *%BLOCKED_KEYWORD%* or other inappropriate words in this community. Review your content and try again."

ModeratedEntityField

The fields and entities you want to moderate.

Field Name	Field Type	Description
`entityName`	string	Required. Indicates the types of user-generated content the moderation rule applies to. Post and comments only apply to content created in groups and user profiles. All feed types, such as polls and links, are supported.
`fieldName`	string	Indicates the field the moderation rule applies to. Note: To moderate feed posts, use `entityName FeedItem` with `fieldName RawBody`. To moderate feed comments, use `entityName FeedComment` with `fieldName RawCommentBody`. The `RawBody` and `RawCommentBody` fields aren't available in any other API.
`keywordList`	KeywordList string	Indicates the keyword list that you want to moderate against.

ModerationRuleType

Required. Indicates the type of rule to run on user-generated content.

Field Name	Field Type	Description
`type`	(enumeration of type string)	Required. Indicates the type of rule to run on user-generated content. Valid values are: • Content • Rate Available in API 39.0 and later.

RateLimitTimePeriod

Required. Indicates the time period that is applied to the rate limit.

Field Name	Field Type	Description
timePeriod	(enumeration of type string)	Required. Indicates the time period that is applied to the rate limit. Valid values are: - Short - Medium Available in API 39.0 and later.

Declarative Metadata Sample Definition

The following is an example of a ModerationRule component.

```
<?xml version="1.0" encoding="UTF-8"?>
<ModerationRule xmlns="http://soap.sforce.com/2006/04/metadata">
   <description>Blocks Bad Word List in posts, comments, Link URLs, titles, and poll
choices.</description>
   <masterLabel>Blocking Rule</masterLabel>
   <action>Block</action>
   <active>true</active>
   <userMessage>You can't use %BLOCKED_KEYWORD% or other inappropriate words in this
community. Review your content and try again.</userMessage>
   <!-- Applies the rule to FeedComment.RawCommentBody (an internal only field), if it
contains words from the keyword list specified -->
   <entitiesAndFields>
     <entityName>FeedComment</entityName>
     <fieldName>RawCommentBody</fieldName>
     <keywordList>community1.badword_list</keywordList>
   </entitiesAndFields>
   <entitiesAndFields>
     <entityName>FeedItem</entityName>
     <fieldName>LinkUrl</fieldName>
     <keywordList>community1.badword_list</keywordList>
   </entitiesAndFields>
  <!-- Applies the rule to FeedItem.RawBody (an internal only field), if it contains words
 from the keyword list specified -->
   <entitiesAndFields>
     <entityName>FeedItem</entityName>
     <fieldName>RawBody</fieldName>
     <keywordList>community1.badword_list</keywordList>
   </entitiesAndFields>
   <entitiesAndFields>
     <entityName>FeedItem</entityName>
     <fieldName>Title</fieldName>
     <keywordList>community1.badword_list</keywordList>
   </entitiesAndFields>
   <entitiesAndFields>
     <entityName>FeedPollChoice</entityName>
     <fieldName>ChoiceBody</fieldName>
     <keywordList>community1.badword_list</keywordList>
   </entitiesAndFields>
</ModerationRule>
```

The following is an example `package.xml` that references the previous definition.

```
<?xml version="1.0" encoding="UTF-8"?>
<Package xmlns="http://soap.sforce.com/2006/04/metadata">
   <types>
     <name>ModerationRule</name>
     <members>community1.blocking_rule</members>
   </types>
   <version>36.0</version>
</Package>
```

Wildcard Support in the Manifest File

This metadata type supports the wildcard character `*` (asterisk) in the `package.xml` manifest file. For information about using the manifest file, see Deploying and Retrieving Metadata with the Zip File.

MyDomainDiscoverableLogin

Represents the configuration settings when the My Domain login page type is Discovery. Login Discovery provides an identity-first login experience, where the login page contains the identifier field only. Based on the identifier entered, a handler determines how to authenticate the user. This type extends the Metadata metadata type and inherits its `fullName` field.

File Suffix and Directory Location

MyDomainDiscoverableLogin components have the suffix `.myDomainDiscoverableLogin` in the `myDomainDiscoverableLogins` folder.

Version

MyDomainDiscoverableLogin components are available in API version 48.0 and later.

Fields

Field Name	Field Type	Description
`apexHandler`	string	Required. The Apex handler class that contains the Discovery authentication logic.
`executeApexHandlerAs`	string	The user who is executing the handler. Requires the Manage User permission.
`usernameLabel`	string	The login prompt when the My Domain login page type is Discovery. This label supports localization with custom labels.

Declarative Metadata Sample Definition

The following is an example of a MyDomainDiscoverableLogin component.

```
<?xml version="1.0" encoding="UTF-8"?>
<MyDomainDiscoverableLogin xmlns="http://soap.sforce.com/2006/04/metadata">
    <apexHandler>MyDomainDiscLoginHandler</apexHandler>
    <executeApexHandlerAs>executeUser@example.com</executeApexHandlerAs>
    <usernameLabel>Enter your email</usernameLabel>
</MyDomainDiscoverableLogin>
```

The following is an example `package.xml` that references the previous definition.

```
<?xml version="1.0" encoding="UTF-8"?>
<Package xmlns="http://soap.sforce.com/2006/04/metadata">
    <types>
        <members>*</members>
        <name>MyDomainDiscoverableLogin</name>
    </types>
    <version>48.0</version>
</Package>
```

Usage

Use this type to access the My Domain Login Discovery Page. This type of login page prompts users to identity themselves with an email address, phone number, or custom identifier. My Domain Login Discovery performs an interview-based login process, where users are prompted to provide identity for authentication. For example, users receive a verification code that they enter to complete the login process.

Wildcard Support in the Manifest File

This metadata type supports the wildcard character `*` (asterisk) in the `package.xml` manifest file. For information about using the manifest file, see Deploying and Retrieving Metadata with the Zip File.

NamedCredential

Represents a named credential, which specifies the URL of a callout endpoint and its required authentication parameters in one definition. A named credential can be specified as an endpoint to simplify the setup of authenticated callouts.

Note: All credentials stored within this entity are encrypted under a framework that is consistent with other encryption frameworks on the platform. Salesforce encrypts your credentials by auto-creating org-specific keys. Credentials encrypted using the previous encryption scheme have been migrated to the new framework.

This type extends the Metadata metadata type and inherits its `fullName` field.

File Suffix and Directory Location

NamedCredential components have the suffix `.namedCredential` and are stored in the `namedCredentials` folder.

Version

NamedCredential components are available in API version 33.0 and later.

Fields

Field Name	Field Type	Description
`awsAccessKey`	string	First part of the access key used to sign programmatic requests to AWS. Use when AWS Signature Version 4 is your authentication protocol. This field is available in API version 46.0 and later.
`awsAccessSecret`	string	The second part of the access key used to sign programmatic requests to AWS. Use when AWS Signature Version 4 is your authentication protocol. This field is available in API version 46.0 and later.
`awsRegion`	string	Specifies which AWS Region the named credential accesses. This field is available in API version 46.0 and later.
`awsService`	string	Specifies which AWS resource the named credential accesses. This field is available in API version 46.0 and later.
`allowMergeFieldsInBody`	boolean	Specifies whether Apex code can use merge fields to populate the HTTP request body with org data when a callout is made. Corresponds to **Allow Merge Fields in HTTP Body** in the user interface. Defaults to `false`. This field is available in API version 41.0 and later.
`allowMergeFieldsInHeader`	boolean	Specifies whether Apex code can use merge fields to populate the HTTP header with org data when a callout is made. Corresponds to **Allow Merge Fields in HTTP Header** in the user interface. Defaults to `false`. This field is available in API version 41.0 and later.
`authProvider`	string	The authentication provider that the AuthProvider component represents.
`authTokenEndpointUrl`	string	The URL where JWTs are exchanged for access tokens. This field is available in API version 46.0 and later.
`certificate`	string	If you specify a certificate, your Salesforce org supplies it when establishing each two-way SSL connection with the external system. The certificate is used for digital signatures, which verify that requests are coming from your Salesforce org.
`endpoint`	string	The URL or root URL of the callout endpoint. Corresponds to **URL** in the user interface.
`generateAuthorizationHeader`	boolean	Specifies whether Salesforce generates an authorization header and applies it to each callout that references the named credential. Corresponds to **Generate Authorization Header** in the user interface. Defaults to `true`. This field is available in API version 41.0 and later.

Field Name	Field Type	Description
jwtAudience	string	External service or other allowed recipients for the JWT. Written as JSON, with a quoted string for a single audience and an array of quoted strings for multiple audiences. Single audience example: “aud1” Multiple audiences example: [“aud1”, “aud2”, “aud3”]
jwtFormulaSubject	string	Formula string calculating the JWT’s Subject. API names and constant strings, in single quotes, can be included. Allows a dynamic Subject unique per user requesting the token. For example, *'User='+$User.Id*. Use this field when principalType is set to PerUser. Corresponds to Per User Subject in the user interface. This field is available in API version 46.0 and later
jwtIssuer	string	Specify who issued the JWT using a case-sensitive string. This field is available in API version 46.0 and later.
jwtSigningCertificate	string	Certificate verifying the JWT’s authenticity to external sites. This field is available in API version 46.0 and later.
jwtTextSubject	string	Static text, without quotes, that specifies the JWT Subject. Use this field when principalType is set to NamedUser. Corresponds to Named Principal Subject in the user interface. This field is available in API version 46.0 and later
jwtValidityPeriodSeconds	string	Specify the number of seconds that the token is valid. This field is available in API version 46.0 and later.
label	string	A user-friendly name for the named credential that appears in the Salesforce user interface, such as in list views.
oauthRefreshToken	string	The OAuth refresh token. Used to obtain a new access token for an end user when a token expires.
oauthScope	string	Specifies the scope of permissions to request for the access token. Corresponds to **Scope** in the user interface.
oauthToken	string	The access token that’s issued by your authorization server.
password	string	The password to be used by your org to access the external system. Ensure that the credentials have adequate privileges to access the external system. Depending on how you set up access, you might need to provide the administrator password.
principalType	External PrincipalType (enumeration of type string)	Determines whether you're using one set or multiple sets of credentials to access the external system. Corresponds to **Identity Type** in the user interface. The valid values are: • Anonymous • PerUser • NamedUser

Field Name	Field Type	Description
protocol	AuthenticationProtocol (enumeration of type string)	The authentication protocol that's required to access the external system. The valid values are: - AwsSig4 - Jwt - JwtExchange - NoAuthentication - Oauth - Password For connections to Amazon Web Services using Signature Version 4, use AwsSig4. For connections using a direct token system, select Jwt. If using an intermediary authorization provider to process JWTs and return access tokens, use JwtExchange. For Simple URL data sources, select NoAuthentication. For cloud-based Files Connect external systems, select Oauth. For on-premises systems, select Password.
username	string	The username to be used by your org to access the external system. Ensure that the credentials have adequate privileges for performing callouts to the external system. Depending on how you set up access, you might need to provide the administrator username.

Declarative Metadata Sample Definition

The following is an example of a NamedCredential component.

```
<?xml version="1.0" encoding="UTF-8"?>
<NamedCredential xmlns="http://soap.sforce.com/2006/04/metadata">
    <allowMergeFieldsInBody>false</allowMergeFieldsInBody>
    <allowMergeFieldsInHeader>false</allowMergeFieldsInHeader>
    <endpoint>https://my_endpoint.example.com</endpoint>
    <generateAuthorizationHeader>true</generateAuthorizationHeader>
    <label>My Named Credential</label>
    <principalType>Anonymous</principalType>
    <protocol>NoAuthentication</protocol>
</NamedCredential>
```

Wildcard Support in the Manifest File

This metadata type supports the wildcard character * (asterisk) in the package.xml manifest file. For information about using the manifest file, see Deploying and Retrieving Metadata with the Zip File.

NavigationMenu

Represents the navigation menu in a community. A navigation menu consists of items that users can click to go to other parts of the community. This type replaces the NavigationLinkSet subtype on Network. NavigationMenu is available in API version 47.0 and later.This type extends the Metadata metadata type and inherits its `fullName` field.

File Suffix and Directory Location

NavigationMenu components have the suffix `.navigationMenu` and are stored in the `navigationMenus` folder.

Version

NavigationMenu components are available in API version 47.0 and later.

Special Access Rules

The MultipleNavigationMenu permission is required.

Fields

Field	Field Type	Description
`container`	string	The name of the navigation menu container.
`containerType`	string	The container type. The options are Network or CommunityTemplateDefinition.
`label`	string	The navigation menu label as it appears in the Experience Builder UI.
`navigationMenuItem`	NavigationMenuItem[]	A list of menu items in a NavigationMenu. Use this object to create, delete, or update menu items in your community's navigation menu.

NavigationMenuItem

Represents a single menu item in the NavigationLinkSet subtype on Network (API version 37.0 to 46.0) or in the NavigationMenu type (API version 47.0 and later). Use this object to create, delete, or update menu items in your community's navigation menu.

Field	Field Type	Description
`defaultListViewId`	string	If the value of the `type` field is `SalesforceObject`, the value is the ID of the default list view for the object.
`label`	string	Required. The text that appears in the navigation menu for this item.

Field	Field Type	Description
`navigationMenuItemBranding`	string	The name of the ContentAsset to be used as branding in the menu item. Available in API version 47.0 and later.
`position`	int	Required. The location of the menu item in the navigation menu.
`publiclyAvailable`	boolean	When set to `true`, gives access to guest users.
`subMenu`	NavigationSubMenu	A list of child menu items. This field is available in API 39.0 and later.
`target`	string	Required if `type` is `ExternalLink`, `InternalLink`, or `SalesforceObject`. If `type` is `ExternalLink` or `InternalLink`, the target is the URL that the link points to. For `ExternalLink`, your entry looks like this: *`http://www.salesforce.com`*. For `InternalLink`, use a relative URL, such as *`/contactsupport`*. If `type` is `MenuLabel` or `NavigationalTopic`, `target` is not used.
`targetPreference`	string	Backed by a picklist that includes preferences for the target field. Valid values are: • `None` • `OpenInExternalTab`—Used for external links to determine whether to open in an external tab.
`type`	string	Required. The type of navigation menu item. Valid values are: • `SalesforceObject`—Available objects include accounts, cases, contacts, and custom objects. • `ExternalLink`—Links to a URL outside of your community. For example, *`http://www.salesforce.com`*. • `InternalLink`—Links to a relative URL inside your community. For example, *`/contactsupport`*. • `MenuLabel`—A parent heading for your navigation menu. See

Field	Field Type	Description
		NavigationSubMenufor how to nest items underneath the menu label. This value is available in API 39.0 and later. • `NavigationalTopic`—A dropdown list with links to the navigational topics in your community. You cannot nest other items of type `MenuLabel` or items of type `NavigationalTopic` under `MenuLabel`.

NavigationSubMenu

A list of child menu items. Only NavigationMenuItem items of type `MenuLabel` can have items in a NavigationSubMenu. Available in API 39.0 and later.

Field	Field Type	Description
`navigationMenuItem`	NavigationMenuItem[]	A list of menu items in a NavigationSubMenu. Use `navigationMenuItem` to create, delete, or update child items under a parent heading.

Declarative Metadata Sample Definition

The following is an example of a NavigationMenu component.

```
<?xml version="1.0" encoding="UTF-8"?>
<NavigationMenu xmlns="http://soap.sforce.com/2006/04/metadata">
    <container>Service</container>
    <containerType>Network</containerType>
    <label>Test Navigation</label>
    <navigationMenuItem>
        <label>Accounts</label>
        <position>1</position>
        <publiclyAvailable>false</publiclyAvailable>
        <target>Account</target>
        <type>SalesforceObject</type>
    </navigationMenuItem>
    <navigationMenuItem>
        <label>External Link</label>
        <menuItemBranding>
            <tileImage>google_image</tileImage>
        </menuItemBranding>
        <position>2</position>
        <publiclyAvailable>false</publiclyAvailable>
        <target>http://google.com</target>
        <targetPreference>OpenExternalLinkInSameTab</targetPreference>
        <type>ExternalLink</type>
    </navigationMenuItem>
    <navigationMenuItem>
        <label>All Objects</label>
        <position>3</position>
        <publiclyAvailable>false</publiclyAvailable>
        <subMenu>
            <navigationMenuItem>
                <label>Leads</label>
                <position>0</position>
                <publiclyAvailable>false</publiclyAvailable>
                <target>Account</target>
                <type>SalesforceObject</type>
            </navigationMenuItem>
        </subMenu>
        <type>MenuLabel</type>
    </navigationMenuItem>
</NavigationMenu>
```

The following is an example `package.xml` that references the previous definition.

```
<?xml version="1.0" encoding="UTF-8"?>
<Package xmlns="http://soap.sforce.com/2006/04/metadata">
  <types>
    <members>*</members>
    <name>NavigationMenu</name>
  </types>
  <version>47.0</version>
</Package>
```

Wildcard Support in the Manifest File

This metadata type supports the wildcard character `*` (asterisk) in the `package.xml` manifest file. For information about using the manifest file, see Deploying and Retrieving Metadata with the Zip File.

Network

Represents a community. Communities are branded spaces for your employees, customers, and partners to connect. You can customize and create communities to meet your business needs, then transition seamlessly between them. Use the Network component for Salesforce Communities. If you want to create zones that contain Chatter Answers and Ideas, use the Community (Zone) component.

This type extends the Metadata metadata type and inherits its `fullName` field.

Declarative Metadata File Suffix and Directory Location

Network components are stored in the `networks` directory of the corresponding package directory. The file name matches the community name, and the extension is `.network`.

Version

This object is available in API version 28.0 and later.

Fields

Field	Field Type	Description
`allowedExtensions`	string	Specifies the types of files allowed in your community. This whitelist of file types lets you control what your community members upload and also prevents spammers from polluting your community with inappropriate files. Available in API version 36.0 and later.
`allowInternalUserLogin`	boolean	Determines whether internal users can log in with their internal credentials on the community login page. Available in API version 40.0 and later.
`allowMembersToFlag`	boolean	Determines whether users in the community can flag posts or comments as inappropriate. Flagged items are sent to a community moderator for review. Available in API version 29.0 and later.
`branding`	Branding	The color scheme, header, and footer used in the community. Deprecated in API version 41.0 and later. Replaced by the NetworkBranding type.
`caseCommentEmailTemplate`	string	Email template used when notifying community members when a case comment has been modified or added to a case. Lightning email templates aren’t packageable. We recommend using a Classic email template.
`changePasswordTemplate`	string	Email template used when notifying a user that their password has been reset. Lightning email templates aren’t packageable. We recommend using a Classic email template.
`communityRoles`	CommunityRoles	Identifies users with Customer, Partner, or Employee roles in a community. Available in API version 41.0 and later.
`description`	string	Description of the community.
`disableReputationRecordConversations`	boolean	When reputation levels are enabled for the community, determines whether to exclude contributions to records when counting points toward reputation levels. Available in API version 41.0 and later.
`emailFooterLogo`	string	The document name of the logo that appears in the footer of community emails. Available in API version 41.0 and later.

Field	Field Type	Description
emailFooterText	string	The text that appears in the footer of community emails. Available in API version 41.0 and later.
emailSenderAddress	string	Required. Email address from which community emails are sent.
emailSenderName	string	Required. Name from which community emails are sent.
enableCustomVFErrorPageOverrides	boolean	Determines whether to use custom Visualforce error pages instead of the default Visualforce error pages. Available in API version 41.0 and later.
enableDirectMessages	boolean	Determines whether community users can send direct messages to start a private conversation with one or more members. Available in API version 41.0 and later.
enableGuestChatter	boolean	Specifies whether guest users can access public Chatter groups in the community without logging in.
enableGuestFileAccess	boolean	Determines whether guest users view asset files shared with the community on publicly accessible pages and login pages. If public access is enabled in Experience Builder at the page or community level, this property is automatically enabled. Available in API version 41.0 and later.
enableGuestMemberVisibility	boolean	Determines if unauthenticated guest users can see the authenticated members of a community (true) or not (false). Available in API version 47.0 and later.
enableInvitation	boolean	Determines whether users can invite others to the community.
enableKnowledgeable	boolean	Determines if community members can see who’s knowledgeable on topics and endorse people for their knowledge on a topic. Available in API version 30.0 and later.
enableNicknameDisplay	boolean	Determines if user nicknames display instead of their first and last names in most places in the community. Set to false by default. Available in API version 32.0 and later.
enablePrivateMessages	boolean	Determines if community members can send and receive private messages. Available in API version 30.0 and later.
enableReputation	boolean	Determines if reputation is calculated and displayed for community members. Available in API version 31.0 and later. If enabled, reputationLevels and reputationPointsRules are used. If no reputationLevels or

Field	Field Type	Description
		reputationPointsRules are not defined in the data file, the default values are used.
enableShowAllNetworkSettings	boolean	Shows settings that are hidden by default based on how the community is set up. Available in API version 41.0 and later.
enableSiteAsContainer	boolean	Determines whether the community uses Site.com pages instead of Visualforce tabs.
enableTalkingAboutStats	boolean	Determines whether community users see how many people are discussing a topic. The number of people discussing the topic appears as the user types the topic and the system gives topic suggestions. Available in API version 41.0 and later.
enableTopicAssignmentRules	boolean	Enables the org to use rules to automatically assign topics to articles in a community. After it's enabled, admins set up rules in a community to map topics to Salesforce Knowledge data categories. This field is available in API version 40.0 and later.
enableTopicSuggestions	boolean	Enables topic suggestions when users write posts. Available in API version 41.0 and later.
enableUpDownVote	boolean	Replaces the option to like a question or answer with the option to upvote or downvote. Available in API version 41.0 and later.
feedChannel	string	Displays the feed of all channel program record or group interactions, including posts, questions, and attachments. This field is available in API version 28.0 and later.
forgotPasswordTemplate	string	Required. The email template used when a user forgets their password. Lightning email templates aren't packageable. We recommend using a Classic email template.
gatherCustomerSentimentData	boolean	Gathers data when a customer looks at articles and cases in communities, for use in the Community 360 feature. This field is available in API version 40.0 and later.
lockoutTemplate	string	The email template used to communicate with users when they get locked out of their org because of too many failed login attempts. Available in API version 43.0 and later. Lightning email templates aren't packageable. We recommend using a Classic email template.

Field	Field Type	Description
`logoutUrl`	string	Specifies the URL that community members are redirected to when they log out from your community. This field is available in API version 28.0 and later.
`maxFileSizeKb`	int	Specifies the maximum file size (in KBs) that members can upload in your community. Available in API version 36.0 and later. Enter a number between 3072 KB and your org's maximum file size. To use the default limit of 2 GB, leave this field empty.
`navigationLinkSet`	NavigationLinkSet	Represents the navigation menu in a community. A navigation menu consists of items that users can click to go to other parts of the community. This field is available in API versions 37.0 to 46.0. In API version 47.0 and later, use the NavigationMenu type instead.
`networkMemberGroups`	NetworkMemberGroups	The profiles and permission sets that have access to the community. Users with these profiles or permission sets are members of the community. Note: If a Chatter customer (from a customer group) is assigned a permission set that is also associated with a community, the Chatter customer isn't added to the community.
`networkPageOverrides`	NetworkPageOverride	The settings in the Administration area (in Community Management or Experience Workspaces) that control which page type Change Password, Forgot Password, Home, and Login each point to. Available in API version 40.0 and later.
`newSenderAddress`	string	Email address that has been entered as the new value for `EmailSenderAddress` but has not been verified yet. After a user has requested to change the sender email address and has successfully responded to the verification email, the `NewSenderAddress` value overwrites the value in `EmailSenderAddress`. This value becomes the email address from which community emails are sent.
`enableMemberVisibility`	boolean	Controls community user visibility on a per-community basis. If `true`, the Community User Visibility preference is enabled for the selected community. Available in API version 45.0 and later.
`picassoSite`	string	Name of the Site.com site linked to the community.
`recommendationAudience`	RecommendationAudience	Creates an audience of new community members, or can be used to manage customized lists of audience members to organize and target recommendations. Available in API version 41.0 and later.

Field	Field Type	Description
`recommendationDefinition`	RecommendationDefinition	Represents a custom recommendation to drive engagement for a community. Targets a specific audience and uses channels to specify a location for the recommendation. Available in API version 41.0 and later.
`reputationLevels`	ReputationLevelDefinitions	The reputation levels assigned to members when they accrue points by performing certain actions.
`reputationPointsRules`	ReputationPointsRules	The points members accrue when they perform certain defined actions.
`selfRegProfile`	string	The profile assigned to users who self register. This value is used only if `selfRegistration` is enabled for the community. Available in API version 29.0 and later.
`selfRegistration`	boolean	Determines whether self-registration is available for the community.
`sendWelcomeEmail`	boolean	Determines whether a welcome email is sent when a new user is added to the community.
`site`	string	Required. The CustomSite associated with the community.
`status`	NetworkStatus[]	Required. Status of the community. Available values are: • `Live`—The community is online and members can access it. • `DownForMaintenance`—The community was previously published but was taken offline. Members with “Create and Set Up Communities” can still access the setup for offline communities regardless of profile or membership. Members are not able to access offline communities, but they still appear in the user interface dropdown as `CommunityName (Offline)`. • `UnderConstruction`—The community has not yet been published. Users with “Create and Set Up Communities” can access communities in this status if their profile is associated with the community. After a community is published, it can never be in this status again.
`tabs`	NetworkTabSet	Required. The tabs that are available in the community. The user that created the community selected these tabs.
`urlPathPrefix`	string	The first part of the path on the site's URL that distinguishes this site from other sites. For example, if your site URL is `mycompany.force.com/partners`, then `partners` is the `urlPathPrefix`.
`verificationTemplate`	string	The email template used to communicate with users when they must verify their identity, for example, when they log

Field	Field Type	Description
		in without a password or from a new device. Available in API version 44.0 and later. Lightning email templates aren’t packageable. We recommend using a Classic email template.
`welcomeTemplate`	string	The email template used when sending welcome emails to new community members. Lightning email templates aren’t packageable. We recommend using a Classic email template.

Branding

Represents the branding and color scheme applied to the community. Available in API version 40.0 and earlier. Replaced by NetworkBranding in API version 41.0 and later.

Field	Field Type	Description
`loginFooterText`	string	The text that appears in the footer of the community login page.
`loginLogo`	string	The logo that appears on the community login page for external users.
`pageFooter`	string	An image that appears on the footer of the community pages. Must be an .html file.
`pageHeader`	string	An image that appears on the header of the community pages. Can be an .html, .gif, .jpg, or .png file.
`primaryColor`	string	The color used for the active tab.
`primaryComplementColor`	string	Font color used with `primaryColor`.
`quaternaryColor`	string	The background color for pages in the community.
`quaternaryComplementColor`	string	Font color used with `quaternaryColor`.
`secondaryColor`	string	The color used for the top borders of lists and tables.
`tertiaryColor`	string	The background color for section headers on edit and detail pages.
`tertiaryComplementColor`	string	Font color used with `tertiaryColor`.
`zeronaryColor`	string	The background color for the header.
`zeronaryComplementColor`	string	Font color used with `zeronaryColor`.

CommunityRoles

The labels used to identify users with Customer, Partner, or Employee roles in a community. Available in API version 41.0 and later.

Field	Field Type	Description
`customerUserRole`	string	The label for the Customer user role in a community.
`employeeUserRole`	string	The label for the Employee user role in a community.
`partnerUserRole`	string	The label for the Partner user role in a community.

NavigationLinkSet

Represents the navigation menu in a community. A navigation menu consists of items that users can click to go to other parts of the community. Available in API versions 37.0 to 46.0. In API version 47.0, use NavigationMenu instead.

Field	Field Type	Description
`navigationMenuItem`	NavigationMenuItem[]	A list of menu items in a NavigationLinkSet. Use this object to create, delete, or update menu items in your community's navigation menu.

NavigationMenuItem

Represents a single menu item in the NavigationLinkSet subtype (API version 37.0 to 46.0) or in the NavigationMenu type (API version 47.0 and later). Use this subtype to create, delete, or update menu items in your community's navigation menu.

Field	Field Type	Description
`defaultListViewId`	string	If the value of the `type` field is `SalesforceObject`, the value is the ID of the default list view for the object.
`label`	string	Required. The text that appears in the navigation menu for this item.
`navigationMenuItemBranding`	string	The name of the ContentAsset to be used as branding in the menu item. Available in API version 47.0 and later.
`position`	int	Required. The location of the menu item in the navigation menu.
`publiclyAvailable`	boolean	When set to `true`, gives access to guest users.

Field	Field Type	Description
`subMenu`	NavigationSubMenu	A list of child menu items. This field is available in API 39.0 and later.
`target`	string	Required if `type` is `ExternalLink`, `InternalLink`, or `SalesforceObject`. If `type` is `ExternalLink` or `InternalLink`, the target is the URL that the link points to. For `ExternalLink`, your entry looks like this: *`http://www.salesforce.com`*. For `InternalLink`, use a relative URL, such as *`/contactsupport`*. If `type` is `MenuLabel` or `NavigationalTopic`, `target` is not used.
`targetPreference`	string	Backed by a picklist that includes preferences for the target field. Valid values are: • `None` • `OpenInExternalTab`—Used for external links to determine whether to open in an external tab.
`type`	string	Required. The type of navigation menu item. Valid values are: • `SalesforceObject`—Available objects include accounts, cases, contacts, and custom objects. • `ExternalLink`—Links to a URL outside of your community. For example, *`http://www.salesforce.com`*. • `InternalLink`—Links to a relative URL inside your community. For example, *`/contactsupport`*. • `MenuLabel`—A parent heading for your navigation menu. See NavigationSubMenufor how to nest items underneath the menu label. This value is available in API 39.0 and later. • `NavigationalTopic`—A dropdown list with links to the navigational topics in your community. You cannot nest other items of type `MenuLabel` or items of type

Field	Field Type	Description
		`NavigationalTopic` under `MenuLabel`.

NavigationSubMenu

A list of child menu items. Only NavigationMenuItem items of type `MenuLabel` can have items in a NavigationSubMenu. Available in API 39.0 and later.

Field	Field Type	Description
`navigationMenuItem`	NavigationMenuItem[]	A list of menu items in a NavigationSubMenu. Use `navigationMenuItem` to create, delete, or update child items under a parent heading.

NetworkMemberGroup

Represents the profiles and permission sets that are assigned to the community. Users with one of the profiles or permission sets are members of the community, unless the user is a Chatter customer (from a customer group).

Field	Field Type	Description
`permissionSet`	string	A permission set that is assigned to the community. Note: If a Chatter customer (from a customer group) is assigned a permission set that is also associated with a community, the Chatter customer isn't added to the community.
`profile`	string	A profile that is part of the community.

NetworkPageOverride

Represents settings in the Administration area (in Community Management or Experience Workspaces) that control which page type the Change Password, Forgot Password, Home, and Login pages each point to.

Field	Field Type	Description
changePasswordPageOverrideSetting	NetworkPageOverrideSetting (enumeration of type string)	Required. Specifies the page type that the Change Password page setting applies to. The valid values are: • Configurable—a Configurable Self-Reg page • Designer—an Experience Builder page • Standard—the default page • VisualForce—a Visualforce page
forgotPasswordPageOverrideSetting	NetworkPageOverrideSetting (enumeration of type string)	Required. Specifies the page type that the Forgot Password page setting applies to. The valid values are: • Configurable—a Configurable Self-Reg page • Designer—an Experience Builder page • Standard—the default page • VisualForce—a Visualforce page
homePageOverrideSetting	NetworkPageOverrideSetting (enumeration of type string)	Required. Specifies the page type that the Community Home page setting applies to. The valid values are: • Configurable—a Configurable Self-Reg page • Designer—an Experience Builder page • Standard—the default page • VisualForce—a Visualforce page
loginPageOverrideSetting	NetworkPageOverrideSetting (enumeration of type string)	Required. Specifies the page type that the Login page setting applies to. The valid values are: • Configurable—a Configurable Self-Reg page • Designer—an Experience Builder page • Standard—the default page • VisualForce—a Visualforce page

Field	Field Type	Description
selfRegProfilePageOverrideSetting	NetworkPageOverrideSetting (enumeration of type string)	Required. Specifies the page type that the Self Registration page setting applies to. The valid values are: - Configurable—a Configurable Self-Reg page - Designer—an Experience Builder page - Standard—the default page - VisualForce—a Visualforce page

RecommendationAudience

Creates an audience of new community members, or can be used to manage customized lists of audience members to organize and target recommendations. Available in API version 41.0 and later.

Field	Field Type	Description
recommendationAudienceDetails	RecommendationAudienceDetail	The specific details of an audience for recommendations.

RecommendationAudienceDetail

The specific details of an audience for recommendations. Available in API version 41.0 and later.

Field	Field Type	Description
audienceCriteriaType	AudienceCriteriaType (enumeration of type string)	The criteria for the recommendation audience type. Values are: - CustomList - MaxDaysInCommunity
audienceCriteriaValue	string	For new member criteria, the maximum number of days since a user became a community member. Null in case of custom list criteria.
setupName	string	Name of the recommendation audience.

RecommendationDefinition

Represents a list of custom recommendations to drive engagement for a community. Available in API version 41.0 and later.

Field	Field Type	Description
`recommendationDefinitionDetails`	RecommendationDefinitionDetail[]	A list of custom recommendations and their details.

RecommendationDefinitionDetail

The specific details of a custom recommendation. Available in API version 41.0 and later.

Field	Field Type	Description
`actionUrl`	string	The URL for the button that lets users act on the recommendation.
`description`	string	An explanation of the recommendation that suggests what users can do.
`linkText`	string	The text label for the button.
`scheduledRecommendations`	ScheduledRecommendation	A list of scheduled recommendations.
`setupName`	string	The name of the recommendation, which appears in Setup.
`title`	string	The title of the recommendation.

ReputationBranding

Branding for the reputation level.

Field	Field Type	Description
`smallImage`	string	Custom image associated with a reputation level. Use files with these extensions: .jpeg, .png, or .gif. Images are stored as documents. If not specified, the default reputation level image is used. Available in API version 32.0 and later.

ReputationLevelDefinitions

Represents reputation levels members can achieve by performing certain defined actions in a community.

Field	Field Type	Description
`level`	ReputationLevel[]	Represents reputation levels.

ReputationLevel

Represents the name and lower value of the reputation level. The application calculates the upper value.

Field	Field Type	Description
`branding`	ReputationBranding[]	Represents any branding associated with the reputation level, specifically, the custom image for the reputation level. This field is optional. If not specified, the default reputation level image is used. Available in API version 32.0 and later.
`label`	string	Name of the reputation level. This field is optional. If not specified, one of the 10 defaults are used. • Level 1 • Level 2 • Level 3 • Level 4 • Level 5 • Level 6 • Level 7 • Level 8 • Level 9 • Level 10
`lowerThreshold`	double	Required. The lower value in the range for this reputation level. For example, if this reputation level is for points 1–50, 1 is the `lowerThreshold`.

ReputationPointsRules

Represents points rules in a community's point system.

Field	Field Type	Description
`pointsRule`	ReputationPointsRule[]	Represents events and their associated points.

ReputationPointsRule

Represents the event and associated point value for a points rule. When a user acts, they accrue the associated points.

Field	Field Type	Description
`eventType`	string	Required. The type of event a member has to perform to get points. The available values are: • `FeedItemWriteAPost` • `FeedItemWriteAComment` • `FeedItemReceiveAComment` • `FeedItemLikeSomething` • `FeedItemReceiveALike` • `FeedItemMentionSomeone` • `FeedItemSomeoneMentionsYou` • `FeedItemShareAPost` • `FeedItemSomeoneSharesYourPost` • `FeedItemPostAQuestion` • `FeedItemAnswerAQuestion` • `FeedItemReceiveAnAnswer` • `FeedItemMarkAnswerAsBest` • `FeedItemYourAnswerMarkedBest` • `FeedItemEndorseSomeoneForKnowledgeOnATopic` • `FeedItemEndorsedForKnowledgeOnATopic`
`points`	int	Required. The number of points a member gets for performing the event. The default number of points per event is: • FeedItemWriteAPost +1 • FeedItemWriteAComment: +1 • FeedItemReceiveAComment: +5 • FeedItemLikeSomething: +1 • FeedItemReceiveALike: +5 • FeedItemMentionSomeone: +1 • FeedItemSomeoneMentionsYou: +5 • FeedItemShareAPost: +1 • FeedItemSomeoneSharesYourPost: +5 • FeedItemPostAQuestion: +1 • FeedItemAnswerAQuestion: +5 • FeedItemReceiveAnAnswer: +5 • FeedItemMarkAnswerAsBest: +5 • FeedItemYourAnswerMarkedBest: +20 • FeedItemEndorseSomeoneForKnowledgeOnATopic: +5 • FeedItemEndorsedForKnowledgeOnATopic: +20

ScheduledRecommendation

Represents a list of scheduled recommendations. Available in API version 41.0 and later.

Field	Field Type	Description
`scheduledRecommendationDetails`	ScheduledRecommendationDetail[]	A list of scheduled recommendations.

ScheduledRecommendationDetail

The specific details of a scheduled recommendation. Available in API version 41.0 and later.

Field	Field Type	Description
`channel`	RecommendationChannel (enumeration of type string)	A way to group recommendations together to determine where they show up in the community. The valid values are: • `DefaultChannel`—The default recommendation channel. Recommendations in the default channel appear in predefined locations, such as directly in the feed in Salesforce mobile web and on the Home and Question Detail pages in communities using the Summer '15 or later version of the Customer Service (Napili) template. • `CustomChannel1`—A custom recommendation channel. Choose where you want recommendations to appear by adding the Recommendations Carousel component to the page in Experience Builder. • `CustomChannel2`—A custom recommendation channel. • `CustomChannel3`—A custom recommendation channel. • `CustomChannel4`—A custom recommendation channel. • `CustomChannel5`—A custom recommendation channel.
`enabled`	boolean	Indicates whether scheduling is enabled. If `true`, the recommendation is enabled and appears in communities. If `false`, recommendations in feeds in Salesforce mobile web aren't removed, but

Field	Field Type	Description
		no new recommendations appear. In communities, disabled recommendations no longer appear.
`rank`	int	The rank of the recommendation within the channel, which determines the order in which it's displayed. The scheduled recommendation is inserted into the position specified by the rank. The rank of all the scheduled recommendations after it is pushed down. If the specified rank is larger than the size of the list, the scheduled recommendation is put at the end of the list. If a rank isn't specified, the scheduled recommendation is put at the end of the list.
`recommendationAudience`	string	The name of the audience for this scheduled recommendation.

NetworkTabSet

Field	Field Type	Description
`customTab`	string	Custom tab that is part of the community.
`defaultTab`	string	The Home tab for the community. When members log in, this tab is the first page they see.
`standardTab`	string	Standard tab that is part of the community.

Declarative Metadata Sample Definition

A sample XML definition of a network.

```
<?xml version="1.0" encoding="UTF-8"?>
<Network xmlns="http://soap.sforce.com/2006/04/metadata">
    <allowMembersToFlag>true</allowMembersToFlag>

<changePasswordTemplate>unfiled$public/CommunityChangePasswordEmailTemplate</changePasswordTemplate>

    <description>Metadata Community</description>
    <emailSenderAddress>admin@networkMetadata.com</emailSenderAddress>
    <emailSenderName>Admin User</emailSenderName>
    <enableInvitation>false</enableInvitation>
    <enableKnowledgeable>true</enableKnowledgeable>
    <enableNicknameDisplay>false</enableNicknameDisplay>
    <enablePrivateMessages>true</enablePrivateMessages>
    <enableReputation>true</enableReputation>
    <enableUpDownVote>true</enableUpDownVote>

<forgotPasswordTemplate>unfiled$public/CommunityForgotPasswordEmailTemplate</forgotPasswordTemplate>

    <networkMemberGroups>
        <permissionSet>Admin</permissionSet>
        <permissionSet>Standard</permissionSet>
        <permissionSet>ReadOnly</permissionSet>
        <profile>Admin</profile>
        <profile>Standard</profile>
        <profile>ReadOnly</profile>
    </networkMemberGroups>
    <recommendationDefinition>
        <recommendationDefinitionDetails>
            <actionUrl>https://www.apple.com/iphone</actionUrl>
            <description>Better specs and high performance for iPhones</description>
            <linkText>iPhone 7</linkText>
            <scheduledRecommendations>
                <scheduledRecommendationDetails>
                    <channel>DefaultChannel</channel>
                    <enabled>false</enabled>
                    <rank>1</rank>
                    <recommendationAudience>New Member Audience</recommendationAudience>
                </scheduledRecommendationDetails>
            </scheduledRecommendations>
            <setupName>Apple iPhone</setupName>
            <title>iPhone7</title>
        </recommendationDefinitionDetails>
        <recommendationDefinitionDetails>
            <actionUrl>https://www.bose.com/qc35</actionUrl>
            <description>New Amazing Noise cancellation Headphones</description>
            <linkText>Bose QC35</linkText>
            <scheduledRecommendations>
                <scheduledRecommendationDetails>
                    <channel>DefaultChannel</channel>
                    <enabled>true</enabled>
                    <rank>2</rank>
                    <recommendationAudience>Custom Audience</recommendationAudience>
                </scheduledRecommendationDetails>
            </scheduledRecommendations>
            <setupName>Bose Headphones</setupName>
            <title>Bose QC35</title>
        </recommendationDefinitionDetails>
    </recommendationDefinition>
    <reputationLevels>
        <level>
            <branding>
                <smallImage>communities_shared
_document_folder/replevel_beginner.png</smallImage>
            </branding>
            <label>Beginner</label>
            <lowerThreshold>0</lowerThreshold>
        </level>
        <level>
            <branding>
                <smallImage>communities_shared
_document_folder/replevel_apprentice.png</smallImage>
            </branding>
            <label>Apprentice</label>
            <lowerThreshold>51</lowerThreshold>
        </level>
        <level>
            <branding>
                <smallImage>communities_shared
_document_folder/replevel_gettingthere.png</smallImage>
            </branding>
            <label>Getting There</label>
            <lowerThreshold>101</lowerThreshold>
        </level>
        <level>
            <branding>
                <smallImage>communities_shared
_document_folder/replevel_skilled.png</smallImage>
            </branding>
            <label>Skilled</label>
            <lowerThreshold>151</lowerThreshold>
        </level>
        <level>
            <branding>
                <smallImage>communities_shared
_document_folder/replevel_expert.png</smallImage>
            </branding>
            <label>Expert</label>
            <lowerThreshold>201</lowerThreshold>
        </level>
        <level>
            <branding>
                <smallImage>communities_shared
_document_folder/replevel_mentor.png</smallImage>
            </branding>
            <label>Mentor</label>
            <lowerThreshold>251</lowerThreshold>
        </level>
        <level>
            <branding>
                <smallImage>communities_shared
_document_folder/replevel_guru.png</smallImage>
            </branding>
            <label>Guru</label>
            <lowerThreshold>301</lowerThreshold>
        </level>
    </reputationLevels>
    <reputationPointsRules>
        <pointsRule>
```

```
            <eventType>FeedItemWriteAPost</eventType>
            <points>5</points>
        </pointsRule>
        <pointsRule>
            <eventType>FeedItemWriteAComment</eventType>
            <points>3</points>
        </pointsRule>
        <pointsRule>
            <eventType>FeedItemReceiveAComment</eventType>
            <points>10</points>
        </pointsRule>
        <pointsRule>
            <eventType>FeedItemLikeSomething</eventType>
            <points>3</points>
        </pointsRule>
        <pointsRule>
            <eventType>FeedItemReceiveALike</eventType>
            <points>5</points>
        </pointsRule>
        <pointsRule>
            <eventType>FeedItemMentionSomeone</eventType>
            <points>5</points>
        </pointsRule>
        <pointsRule>
            <eventType>FeedItemSomeoneMentionsYou</eventType>
            <points>10</points>
        </pointsRule>
        <pointsRule>
            <eventType>FeedItemShareAPost</eventType>
            <points>5</points>
        </pointsRule>
        <pointsRule>
            <eventType>FeedItemSomeoneSharesYourPost</eventType>
            <points>10</points>
        </pointsRule>
    </reputationPointsRules>
    <selfRegistration>false</selfRegistration>
    <sendWelcomeEmail>true</sendWelcomeEmail>
    <site>Network_11</site>
    <status>UnderConstruction</status>
    <tabs>
        <defaultTab>Chatter</defaultTab>
        <standardTab>Chatter</standardTab>
        <standardTab>Account</standardTab>
        <standardTab>Campaign</standardTab>
        <standardTab>Case</standardTab>
        <standardTab>Console</standardTab>
        <standardTab>Contact</standardTab>
        <standardTab>Contract</standardTab>
        <standardTab>Dashboard</standardTab>
        <standardTab>JigsawSearch</standardTab>
        <standardTab>File</standardTab>
        <standardTab>CollaborationGroup</standardTab>
        <standardTab>home</standardTab>
        <standardTab>Idea</standardTab>
        <standardTab>Lead</standardTab>
        <standardTab>Opportunity</standardTab>
        <standardTab>Product2</standardTab>
        <standardTab>UserProfile</standardTab>
        <standardTab>report</standardTab>
        <standardTab>Solution</standardTab>
    </tabs>
    <urlPathPrefix>network1</urlPathPrefix>
    <welcomeTemplate>unfiled$public/CommunityWelcomeEmailTemplate</welcomeTemplate>
</Network>
```

Wildcard Support in the Manifest File

This metadata type supports the wildcard character `*` (asterisk) in the `package.xml` manifest file. For information about using the manifest file, see Deploying and Retrieving Metadata with the Zip File.

SEE ALSO:

Community (Zone)

NetworkBranding

Represents the branding and color scheme applied to the login pages of a community. (Salesforce Communities are represented by the Network component.)

This type extends the MetadataWithContent type and inherits its `content` and `fullName` fields.

Note: For branding properties that apply to Lightning communities, see BrandingSet.

Declarative Metadata File Suffix and Directory Location

NetworkBranding components have the suffix `.networkBranding` and are stored in the `networkBranding` folder.

Version

This object is available in API version 41.0 and later. It replaces the Branding subtype in the Network component.

Fields

Field	Field Type	Description
`loginBackgroundImageUrl`	string	The path to the image URL that appears as the background on the community's login page. This URL can be fixed, dynamic, or an uploaded image. A dynamic URL contains the experience ID parameter, `{expid}`, and is resolved dynamically at runtime.

Field	Field Type	Description
`loginFooterText`	string	The text that appears in the footer of the community login page.
`loginLogo`	string	The logo that appears on the community login page for external users.
`loginLogoName`	string	The name of the logo that appears on the community login page for external users.
`loginPrimaryColor`	string	The background color of the Login button. Available in API version 42.0 and later.
`loginQuaternaryColor`	string	The background color for the community's login page.
`loginRightFrameUrl`	string	The path to the content of the right frame of the community login page. This URL can be either fixed or dynamic. A dynamic URL contains the experience ID parameter, `{expid}`. If the URL contains `{expid}`, the URL is resolved dynamically at runtime depending on the parameter's value.
`network`	string	The name of the community associated with the branding.
`pageFooter`	string	An image that appears on the footer of the community pages. Must be an .html file.
`pageHeader`	string	An image that appears on the header of the community pages. Can be an .html, .gif, .jpg, or .png file.
`primaryColor`	string	Required. The color used for the active tab.
`primaryComplementColor`	string	Required. Font color used with `primaryColor`.
`quaternaryColor`	string	Required. The background color for pages in the community.
`quaternaryComplementColor`	string	Required. Font color used with `quaternaryColor`.
`secondaryColor`	string	Required. The color used for the top borders of lists and tables.
`staticLogoImageUrl`	string	The path to the logo that appears on the community's login page. This URL can be fixed, dynamic, or an uploaded image. A dynamic URL contains the experience ID parameter, `{expid}`. If the URL contains `{expid}`, the URL is resolved dynamically

Field	Field Type	Description
		at runtime depending on the parameter's value.
`tertiaryColor`	string	Required. The background color for section headers on edit and detail pages.
`tertiaryComplementColor`	string	Required. Font color used with `tertiaryColor`.
`zeronaryColor`	string	Required. The background color for the header.
`zeronaryComplementColor`	string	Required. Font color used with `zeronaryColor`.

Declarative Metadata Sample Definition

A sample XML definition of network branding.

```
<?xml version="1.0" encoding="UTF-8"?>
<NetworkBranding xmlns="http://soap.sforce.com/2006/04/metadata">
    <loginFooterText>salesforce.com</loginFooterText>
    <loginLogo>Communities_Shared_Document_Folder/header2_png.png</loginLogo>
    <loginLogoName>header2.png</loginLogoName>

<loginBackgroundImageUrl>http://identitycms.herokuapp.com/promo-background.jpg</loginBackgroundImageUrl>

    <loginQuaternaryColor>#B1BAC1</loginQuaternaryColor>
    <loginRightFrameUrl>http://www.example.com/test</loginRightFrameUrl>
    <network>Network 1</network>
    <pageFooter>Branding/footer_html.html</pageFooter>
    <pageHeader>Branding/header_Image.jpg</pageHeader>
    <primaryColor>#AF5800</primaryColor>
    <primaryComplementColor>#FFFFFF</primaryComplementColor>
    <quaternaryColor>#286FB8</quaternaryColor>
    <quaternaryComplementColor>#FFFFFF</quaternaryComplementColor>
    <secondaryColor>#000000</secondaryColor>
    <tertiaryColor>#FFFFFF</tertiaryColor>
    <tertiaryComplementColor>#222222</tertiaryComplementColor>
    <zeronaryColor>#0A3764</zeronaryColor>
    <zeronaryComplementColor>#FFFFFF</zeronaryComplementColor>
</NetworkBranding>
```

Wildcard Support in the Manifest File

This metadata type supports the wildcard character `*` (asterisk) in the `package.xml` manifest file. For information about using the manifest file, see Deploying and Retrieving Metadata with the Zip File.

NotificationTypeConfig

Represents the metadata associated with org-level notification settings for standard and custom notification types. This type extends the Metadata metadata type and inherits its `fullName` field.

File Suffix and Directory Location

NotificationTypeConfig components have the suffix `.config` and are stored in the `notificationTypeConfig` folder.

Version

NotificationTypeConfig components are available in API version 48.0 and later.

Fields

Field Name	Field Type	Description
`notificationTypeSettings`	NotificationTypeSettings on page 696[]	An array of delivery settings for an org's notification types.

NotificationTypeSettings

Represents the delivery settings for a standard or custom notification type.

Field Name	Field Type	Description
`notificationType`	string	Required. Specifies a notification type's API name. For standard notification types, this is the predefined API name of the standard notification type. For custom notification types, this is the API name of the custom notification type. If a custom notification type was installed with a managed package, it includes the namespace prefix. Retrieve NotificationTypeConfig to see the API names of the notification types available in your org.
`appSettings`	AppSettings on page 696[]	An array of settings for the connected apps supported for a notification type.
`notificationChannels`	NotificationChannels on page 697	Defines the delivery channels for a notification type.

AppSettings

Represents the settings for the connected apps supported for a notification type.

Field Name	Field Type	Description
connectedAppName	string	Required. Specifies the API name of a connected app. If a connected app was installed with a managed package, it includes the namespace prefix. Retrieve NotificationTypeConfig to see the API names of the connected apps supported for a notification type.
enabled	boolean	Indicates whether a connected app is enabled (true) or not (false) for the notification type.

NotificationChannels

Represents the settings for the delivery channels for a notification type.

Field Name	Field Type	Description
desktopEnabled	boolean	Indicates whether desktop notifications are enabled (true) or not (false).
mobileEnabled	boolean	Indicates whether mobile notifications are enabled (true) or not (false).

Declarative Metadata Sample Definition

The following is an example of a NotificationTypeConfig component.

```
<NotificationTypeConfig xmlns="http://soap.sforce.com/2006/04/metadata">
    <notificationTypeSettings>
        <notificationType>chatter_mention</notificationType>
        <notificationChannels>
            <desktopEnabled>false</desktopEnabled>
            <mobileEnabled>true</mobileEnabled>
        </notificationChannels>
        <appSettings>
            <connectedAppName>Datawatch</connectedAppName>
            <enabled>false</enabled>
        </appSettings>
            <appSettings>
            <connectedAppName>package2__ConnectedApp2</connectedAppName>
            <enabled>true</enabled>
        </appSettings>
    </notificationTypeSettings>
   <notificationTypeSettings>
        <notificationType>namespace__Custom_Notification</notificationType>
        <notificationChannels>
            <desktopEnabled>true</desktopEnabled>
            <mobileEnabled>true</mobileEnabled>
        </notificationChannels>
        <appSettings>
            <connectedAppName>namespace__Connected_App</connectedAppName>
            <enabled>false</enabled>
        </appSettings>
        <appSettings>
```

```
            <connectedAppName>namespace2__ConnectedApp2</connectedAppName>
            <enabled>true</enabled>
        </appSettings>
    </notificationTypeSettings>
</NotificationTypeConfig>
```

The following is an example of a package manifest used to retrieve all the available notification settings for an organization, using a wildcard:

```
<?xml version="1.0" encoding="UTF-8"?>
<Package xmlns="http://soap.sforce.com/2006/04/metadata">
    <types>
        <members>*</members>
        <name>NotificationTypeConfig</name>
    </types>
    <version>48.0</version>
</Package>
```

Wildcard Support in the Manifest File

This metadata type supports the wildcard character `*` (asterisk) in the `package.xml` manifest file. For information about using the manifest file, see Deploying and Retrieving Metadata with the Zip File.

OauthCustomScope

Represents a permission defining the protected data that a connected app can access from an external entity when Salesforce is the OAuth authorization provider.

File Suffix and Directory Location

OauthCustomScope components have the suffix `.oauthcustomscope` and are stored in the `oauthcustomscopes` directory.

Version

OAuth custom scopes are available in API version 46.0 and later.

Special Access Rules

You must have the “Manage Connected Apps” permission to access this object.

Fields

Field Name	Field Type	Description
`description`	string	Required. The description of the permission provided to the connected app by the scope. The custom scope’s description must be unique, can only include alphanumeric characters, and can be up to 60 characters long.

Field Name	Field Type	Description
		You can enter a custom label in place of a description. An advantage of using a custom label is that you can maintain reusable text in a single location and translate the text into multiple languages. See Custom Labels. Note: The description formatting requirements that apply to custom scopes also apply to custom labels.
developerName	string	Required. Use when referring to the OAuth custom scope from a program.
isProtected	boolean	Required. Indicates whether this component is protected () or not (false). Protected components cannot be linked to or referenced by components created in the installing org.
isPublic	boolean	Indicates whether the object is included in the connected app's OpenID Connect discovery endpoint. The default setting is false. For more information, see OpenID Connect Discovery Endpoint.
masterLabel	string	Required. The master label for the custom scope record. This label must be unique and begin with a letter. It can include only alphanumeric characters and underscores. It can't contain spaces.

Declarative Metadata Sample Definition

The following is an example of an OAuthCustomScope component. In this example, basicScope is the name of custom scope entity being retrieved. You can also enter an asterisk (*) to retrieve all the members of this entity type.

```
<?xml version="1.0" encoding="UTF-8"?>
<OauthCustomScope xmlns="http://soap.sforce.com/2006/04/metadata">
<description>Example of a basic custom scope</description>
<developerName>basicScope</developerName>
<isProtected>false</isProtected>
<isPublic>true</isPublic>
<masterLabel>basicScope</masterLabel>
</OauthCustomScope>
```

The following is an example package.xml that references the previous definition.

```
<?xml version="1.0" encoding="UTF-8"?>
<Package xmlns="http://soap.sforce.com/2006/04/metadata">
<types>
<members>basicScope</members>
<name>OauthCustomScope</name>
</types>
<version>46.0</version>
</Package>
```

Usage

An OAuth custom scope tells an external entity about a connected app's permissions to access protected data. The OAuth custom scope you create in your Salesforce org corresponds to the same custom scope defined in your external entity and assigned to the resource.

For example, you define an Order Status custom scope in your external entity that allows access to customer order status data in your order system's API. In Salesforce, you create an OAuth custom scope that you also name Order Status. You assign this custom scope to the connected app requesting access to the order status API. When the external entity receives the connected app's request to access a customer's order status, it validates the connected app's access token and Order Status scope. With a successful validation, the app can access the customer order status information in the order system's API.

Wildcard Support in the Manifest File

This metadata type supports the wildcard character `*` (asterisk) in the `package.xml` manifest file. For information about using the manifest file, see Deploying and Retrieving Metadata with the Zip File.

Package

Specifies which metadata components to retrieve as part of a `retrieve()` call or defines a package of components.

Name	Type	Description
`apiAccessLevel`	APIAccessLevel (enumeration of type string)	Package components have access via dynamic Apex and the API to standard and custom objects in the organization where they are installed. Administrators who install packages may wish to restrict this access after installation for improved security. The valid values are: - Unrestricted—Package components have the same API access to standard objects as the user who is logged in when the component sends a request to the API. - Restricted—The administrator can select which standard objects the components can access. Further, the components in restricted packages can only access custom objects in the current package if the user's permissions allow access to them. For more information, see "About API and Dynamic Apex Access in Packages" in the Salesforce online help.
`description`	string	A short description of the package.
`fullName`	string	The package name used as a unique identifier for API access. The `fullName` can contain only underscores and alphanumeric characters. It must be unique, begin with a letter, not include spaces, not end with an underscore, and not contain two consecutive underscores. This field is inherited from the Metadata component.
`namespacePrefix`	string	The namespace of the developer organization where the package was created.

Name	Type	Description
objectPermissions	ProfileObjectPermissions[]	Indicates which objects are accessible to the package, and the kind of access available (create, read, update, delete).
packageType	string	Reserved for future use.
postInstallClass	string	The name of the Apex class that specifies the actions to execute after the package has been installed or upgraded. The Apex class must be a member of the package and must implement the Apex InstallHandler interface. In patch upgrades, you can't change the class name in this field but you can change the contents of the Apex class. The class name can be changed in major upgrades. This field is available in API version 24.0 and later.
setupWeblink	string	The weblink used to describe package installation.
types	PackageTypeMembers[]	The type of component being retrieved.
uninstallClass	string	The name of the Apex class that specifies the actions to execute after the package has been uninstalled. The Apex class must be a member of the package and must implement the Apex UninstallHandler interface. In patch upgrades, you can't change the class name in this field but you can change the contents of the Apex class. The class name can be changed in major upgrades. This field is available in API version 25.0 and later.
version	string	Required. The version of the component type.

PackageTypeMembers

Use to specify the name and type of components to be retrieved in a package.

Name	Type	Description
members	string	One or more named components, or the wildcard character (*) to retrieve all metadata components of the type specified in the <name> element. To retrieve a standard object, specify it by name. For example, <members>Account</members> retrieves the standard Account object.
name	string	The type of metadata component to be retrieved. For example, <name>CustomObject</name> retrieves one or more custom objects as specified in the <members> element.

Wildcard Support in the Manifest File

This metadata type doesn't support the wildcard character `*` (asterisk) in the `package.xml` manifest file. For information about using the manifest file, see Deploying and Retrieving Metadata with the Zip File.

SEE ALSO:

Sample package.xml Manifest Files

PathAssistant

Represents Path records.This type extends the Metadata metadata type and inherits its `fullName` field.

Note the following when working with PathAssistant:

- Only one path can be created per record type for each object, including __Master__ record type.
- Rich text guidance information cannot be retrieved or deployed from or to translation workbench.
- The preference does not need to be on to retrieve or deploy PathAssistant.

File Suffix and Directory Location

PathAssistant components have the suffix `.pathAssistant` and are stored in the `pathAssistants` folder.

Version

PathAssistant components are available in API version 34.0 and later.

Fields

Field Name	Field Type	Description
`active`	boolean	Indicates whether the path is active (`true`) or not (`false`).
`entityName`	string	Required. The entity name. This is hard coded for Opportunity, Lead, and Quote. For a custom object, this field must be specified and should be the name of the custom object. This field is not updateable.
`fieldName`	string	Required. The field name. This is hard coded for StageName and Status. For a custom object, this field must be specified and should be the name of the picklist field that determines the steps in the path. This field is not updateable.
`masterLabel`	string	Required. The master label of the path.
`pathAssistantSteps`	PathAssistantStep[] on page 703	List of all the steps that have been configured with fields and guidance information. Note that a missing step in the .xml file means it has not been configured, not that it doesn't exist.
`recordTypeName`	string	Required. The name of the record type associated with the path. This field is not updateable.

PathAssistantStep

Represents the steps or stages in a Path.

Field Name	Field Type	Description
`fieldNames`	string	All the fields in `entityName` that will display in this step.
`info`	string	The guidance information displayed in this step.
`picklistValueName`	string	Required. The picklist value associated with the step.

Declarative Metadata Sample Definition

The following is an example of a PathAssistant component.

```
<?xml version="1.0" encoding="UTF-8"?>
<PathAssistant xmlns="http://soap.sforce.com/2006/04/metadata">
    <active>true</active>
    <entityName>Opportunity</entityName>
    <fieldName>StageName</fieldName>
    <masterLabel>Test Path</masterLabel>
    <pathAssistantSteps>
        <fieldNames>Amount</fieldNames>
        <fieldNames>CloseDate</fieldNames>
        <info>Some Text</info>
        <picklistValueName>Id. Decision Makers</picklistValueName>
    </pathAssistantSteps>
    <pathAssistantSteps>
        <fieldNames>Amount</fieldNames>
        <fieldNames>CloseDate</fieldNames>
        <info>Some Text</info>
        <picklistValueName>Proposal/Price Quote</picklistValueName>
    </pathAssistantSteps>
    <recordTypeName>Test_Record_Type</recordTypeName>
</PathAssistant>
```

The following is an example `package.xml` that references the previous definition.

```
<?xml version="1.0" encoding="UTF-8"?>
<Package xmlns="http://soap.sforce.com/2006/04/metadata">
  <types>
        <members>Opportunity.Test_Busines_Process</members>
        <name>BusinessProcess</name>
    </types>
    <types>
        <members>Opportunity.StageName</members>
        <members>Lead.LeadSource</members>
        <members>Opportunity.Type</members>
        <name>CustomField</name>
    </types>
    <types>
        <members>Test_Path</members>
        <name>PathAssistant</name>
    </types>
    <types>
        <members>Opportunity.Test_Record_Type</members>
        <name>RecordType</name>
    </types>
    <types>
        <members>PathAssistant</members>
        <name>Settings</name>
    </types>
    <version>48.0</version>
</Package>
```

Wildcard Support in the Manifest File

This metadata type supports the wildcard character `*` (asterisk) in the `package.xml` manifest file. For information about using the manifest file, see Deploying and Retrieving Metadata with the Zip File.

PaymentGatewayProvider

Represents the metadata associated with a payment gateway provider. This type extends the Metadata metadata type and inherits its `fullName` field.

File Suffix and Directory Location

PaymentGatewayProvider components have the suffix `paymentGatewayProvider` and are stored in the `paymentGatewayProviders` folder.

Version

PaymentGatewayProvider components are available in API version 48.0 and later.

Special Access Rules

To access PaymentGatewayProvider, you must have a Salesforce Order Management license with the PaymentPlatform org permission activated.

Fields

Field Name	Field Type	Description
`apexAdapter`	string	The Apex adapter class name for your payment gateway. This field is unique within your organization.
`comments`	string	Users can add comments to provide additional details about a record. Maximum of 1000 characters.

Field Name	Field Type	Description
`idempotencySupported`	IdempotencySupportStatus (enumeration of type String)	Required. Defines whether the payment gateway ignores duplicate payment gateway calls (`Yes`) or whether it processes duplicate gateway calls (`No`). • `Yes` • `No`
`isProtected`	boolean	Indicates whether the payment gateway provider is protected (`true`) or not protected (`false`). Protected components can't be linked to or referenced by components created in a subscriber org. A developer can delete a protected component in a future release without worrying about failing installations. However, once a component is marked as unprotected and is released globally, the developer can't delete it.
`masterLabel`	string	Required. The label of this payment gateway provider record.

Declarative Metadata Sample Definition

The following is an example of a PaymentGatewayProvider component.

```
<PaymentGatewayProvider xmlns="http://soap.sforce.com/2006/04/metadata">
    <apexAdapter>SalesforceAdapter</apexAdapter>
    <idempotencySupported>Yes</idempotencySupported>
    <masterLabel>SalesforceAdapter</masterLabel>
    <isProtected>true</isProtected>
    <comments>Comments</comments>
</PaymentGatewayProvider>
```

The following is an example `package.xml` that references the previous definition.

```
<Package xmlns="http://soap.sforce.com/2006/04/metadata">
    <types>
        <members>*</members>
        <name>PaymentGatewayProvider</name>
    </types>
    <version>48.0</version>
</Package>
```

Wildcard Support in the Manifest File

This metadata type supports the wildcard character `*` (asterisk) in the `package.xml` manifest file. For information about using the manifest file, see Deploying and Retrieving Metadata with the Zip File.

PermissionSet

Represents a set of permissions that's used to grant additional access to one or more users without changing their profile or reassigning profiles. You can use permission sets to grant access but not to deny access.

This type extends the Metadata metadata type and inherits its `fullName` field.

Important: In API version 40.0 and later, when you retrieve or deploy permission set metadata, all content exposed in Metadata API for the permission sets is included. The metadata includes Apex associated with the permission set, CRUD, and so on.

In API version 39.0 and earlier, retrieving or deploying permission set metadata returns only app and system permissions assigned to the permission set. Junction metadata (such as Apex, CRUD) are included only if the metadata for the related component is also included in the package definition.

In API version 29.0 and later, you can retrieve and deploy access settings for the following managed components in profiles and permission sets:

- Apex classes
- Apps
- Custom field permissions
- Custom object permissions
- Custom tab settings
- External data sources
- Record types
- Visualforce pages

For more information, see Managed Component Access in Sample `package.xml` Manifest Files on page 23.

Declarative Metadata File Suffix and Directory Location

Permission sets are stored in the `permissionsets` directory. The file name matches the permission set API name and the extension is `.permissionset`. For example, a permission set with the name *User_Management_Perms* is stored in `permissionsets/User_Management_Perms.permissionset`.

Version

Permission sets are available in API version 22.0 and later.

Special Access Rules

As of Spring '20 and later, only users who have one of these permissions can access this object:

- View Setup and Configuration
- Manage Session Permission Set Activations
- Manage Users
- Assign Permission Sets
- Manage External Users
- Manage Customer Users

Alternatively, the user is a Delegated External User Administrator.

Fields

Field	Field Type	Description
`applicationVisibilities`	PermissionSetApplicationVisibility[]	Indicates which apps are visible to users assigned to this permission set. Available in API version 29.0 and later. In API version 29.0, this field supports custom apps only. In API version 30.0 and later, this field supports both standard and custom apps.
`classAccesses`	PermissionSetApexClassAccess[]	Indicates which top-level Apex classes have methods that users assigned to this permission set can execute. Available in API version 23.0 and later.
`customMetadataTypeAccesses`	PermissionSetCustomMetadataTypeAccess[]	Indicates the custom metadata types that are read-accessible to a user assigned to this permission set. Available in API version 47.0 and later.
`customPermissions`	PermissionSetCustomPermissions[]	Indicates which custom permissions are available to users assigned to this permission set. Available in API version 31.0 and later.
`customSettingAccesses`	PermissionSetCustomSettingAccesses[]	Indicates the custom settings that are read-accessible to a user assigned to this permission set. Available in API version 47.0 and later.
`description`	string	The permission set description. Limit: 255 characters.
`externalDataSourceAccesses`	PermissionSetExternal DataSourceAccess[]	Indicates which data sources with identity type of `Per User` are available to users assigned to this permission set. Available in API version 27.0 and later.
`fieldPermissions`	PermissionSetFieldPermissions[]	Indicates which fields are accessible to a user assigned to this permission set, and the kind of access available (readable or editable). Available in API version 23.0 and later.
`hasActivationRequired`	boolean	Indicates whether the permission set requires an associated active session (`true`) or not (`false`). Available in API version 37.0 and later.
`label`	string	Required. The permission set label. Limit: 80 characters.
`license`	string	Either the related permission set license or the user license associated with this permission set. Available in API version 38.0 and later. Use this field instead of `userLicense`, which is deprecated and only available up to API Version 37.0.
`objectPermissions`	PermissionSetObjectPermissions[]	Indicates the objects that are accessible to a user assigned to this permission set, and the kind of access available (create, read, edit, delete, and so on). Available in API version 23.0 and later.

Field	Field Type	Description
`pageAccesses`	PermissionSetApexPageAccess[]	Indicates which Visualforce pages that users assigned to this permission set can execute. Available in API version 23.0 and later.
`recordTypeVisibilities`	PermissionSetRecordTypeVisibility[]	Indicates which record types are visible to users assigned to this permission set. Available in API version 29.0 and later. This field is never retrieved or deployed for inactive record types.
`tabSettings`	PermissionSetTabSetting[]	Indicates the tab visibility settings for this permission set. Available in API version 26.0 and later.
`userLicense`	string	Deprecated. The user license for the permission set. A user license determines the baseline of features that the user can access. Every user must have exactly one user license. Available up to API version 37.0. In API version 38.0 and later, use `license`.
`userPermissions`	PermissionSetUserPermission[]	Specifies an app or system permission (such as "API Enabled") and whether it's enabled for this permission set. In API version 28.0 and earlier, this field retrieves all user permissions, enabled or disabled. In API version 29.0 and later, this field retrieves only enabled user permissions.

PermissionSetApplicationVisibility

PermissionSetApplicationVisibility determines whether an app is visible to a user assigned to this permission set.

Field Name	Field Type	Description
`application`	string	Required. The app name.
`visible`	boolean	Required. Indicates whether this app is visible to users assigned to this permission set (`true`) or not (`false`).

PermissionSetApexClassAccess

PermissionSetApexClassAccess represents the Apex class access for users assigned to a permission set.

Field	Field Type	Description
`apexClass`	string	Required. The Apex class name.
`enabled`	boolean	Required. Indicates whether users assigned to this permission set can execute methods in the top-level class (`true`) or not (`false`).

PermissionSetCustomMetadataTypeAccess

PermissionSetCustomMetadataTypeAccess represents the custom metadata type access for users assigned to a permission set. Available in API version 47.0 and later.

Field	Field Type	Description
`enabled`	boolean	Required. Indicates whether the records for this custom metadata type are readable (`true`) or not (`false`).
`name`	string	Required. The custom metadata type name.

PermissionSetCustomPermissions

PermissionSetCustomPermissions represents the custom permissions access for users assigned to a permission set. Only enabled custom permissions are retrieved.

Field Name	Field Type	Description
`enabled`	boolean	Required. Indicates whether the custom permission is enabled (`true`) or not (`false`).
`name`	string	Required. The custom permission name.

PermissionSetCustomSettingAccesses

PermissionSetCustomSettingAccesses represents the custom setting access for users assigned to a permission set. Available in API version 47.0 and later.

Field	Field Type	Description
`enabled`	boolean	Required. Indicates whether the records for this custom setting are readable (`true`) or not (`false`).
`name`	string	Required. The custom setting name.

PermissionSetExternalDataSourceAccess

PermissionSetExternalDataSourceAccess represents the data source access for users with identity type of `Per User`. Available in API version 27.0 and later.

Field	Field Type	Description
`enabled`	boolean	Required. Indicates whether the data source is enabled (`true`) or not (`false`).
`externalDataSource`	string	The name of the external data source.

PermissionSetFieldPermissions

PermissionSetFieldPermissions represents the field permissions for users assigned to a permission set. In API version 30.0 and later, permissions for required fields can't be retrieved or deployed.

Note: As of API version 38.0, you can change field permissions to make a field editable using the Metadata API for fields that you can't change through the user interface. For example, you can deploy `Asset.ProductCode` as an editable field even though you can't through the user interface.

Field	Field Type	Description
`editable`	boolean	Required. Indicates whether the field can be edited by the users assigned to this permission set (`true`) or not (`false`).
`field`	string	Required. The API name of the field (such as `Warehouse__c.Description__c`).
`readable`	boolean	Indicates whether the field can be read by the users assigned to this permission set (`true`) or not (`false`).

PermissionSetObjectPermissions

PermissionSetObjectPermissions represents the object permissions for a permission set. Use one of these elements for each permission.

Field	Field Type	Description
`allowCreate`	boolean	Required. Indicates whether the object referenced by the `object` field can be created by the users assigned to this permission set (`true`) or not (`false`).
`allowDelete`	boolean	Required. Indicates whether the object referenced by the `object` field can be deleted by the users assigned to this permission set (`true`) or not (`false`).
`allowEdit`	boolean	Required. Indicates whether the object referenced by the `object` field can be edited by the users assigned to this permission set (`true`) or not (`false`).
`allowRead`	boolean	Required. Indicates whether the object referenced by the `object` field can be viewed by the users assigned to this permission set (`true`) or not (`false`).
`modifyAllRecords`	boolean	Required. Indicates whether the object referenced by the `object` field can be viewed, edited, or deleted by the users assigned to this permission set (`true`) or not (`false`), regardless of the sharing settings for the object. This includes private records (records with no parent object). This is similar to the "Modify All Data" user permission, but limited to the individual object level.
`object`	string	Required. The API name of the object (such as `Warehouse__c`).
`viewAllRecords`	boolean	Required. Indicates whether the object referenced by the `object` field can be viewed by the users assigned to this permission set

Field	Field Type	Description
		(`true`) or not (`false`), regardless of the sharing settings for the object. This includes private records (records with no parent object). The viewAllRecords field is similar to the "View All Data" user permission but limited to the individual object level.

PermissionSetApexPageAccess

PermissionSetApexPageAccess represents the Visualforce page access for users assigned to a permission set.

Field	Field Type	Description
`apexPage`	string	Required. The Visualforce page name.
`enabled`	boolean	Required. Indicates whether users assigned to this permission set can execute the Visualforce page (`true`) or not (`false`).

PermissionSetRecordTypeVisibility

PermissionSetRecordTypeVisibility represents the visibility of record types for this permission set.

Field	Field Type	Description
`recordType`	string	Required. The record type name, for example `Account.MyRecordType`.
`visible`	boolean	Required. Indicates whether the record type is visible to users assigned to this permission set (`true`) or not (`false`).

PermissionSetTabSetting

PermissionSetTabSetting represents the tab settings for a permission set.

Field	Field Type	Description
`tab`	string	Required. The tab name.
`visibility`	PermissionSetTabVisibility (enumeration of type string)	Required. Indicates the visibility settings for the tab. Valid values are: • `Available`—The tab is available on the All Tabs page. Individual users can customize their display to make the tab visible in any app. • `None`—The tab isn't available on the All Tabs page or visible in any apps. • `Visible`—The tab is available on the All Tabs page and appears in the visible tabs for its associated app. Individual

Field	Field Type	Description
		users can customize their display to hide the tab or make it visible in other apps.

PermissionSetUserPermission

In API version 28.0 and earlier, PermissionSetUserPermission represents an app or system permission for a permission set. In API version 29.0 and later, this field retrieves only enabled user permissions. Use one of these elements for each permission.

Field	Field Type	Description
`enabled`	boolean	Required. Indicates whether the permission is enabled (`true`) or disabled (`false`).
`name`	string	Required. The name of the permission.

Declarative Metadata Sample Definition

When adding or changing a permission set, you don't need to include all permissions—you only need to include the permissions you're adding or changing.

```
<?xml version="1.0" encoding="UTF-8"?>
<PermissionSet xmlns="http://soap.sforce.com/2006/04/metadata">
    <description>Grants all rights needed for an HR administrator to manage
employees.</description>
    <label>HR Administration</label>
    <userLicense>Salesforce</userLicense>
    <applicationVisibilities>
        <application>JobApps__Recruiting</application>
        <visible>true</visible>
    </applicationVisibilities>
    <userPermissions>
        <enabled>true</enabled>
        <name>APIEnabled</name>
    </userPermissions>
    <objectPermissions>
        <allowCreate>true</allowCreate>
        <allowDelete>true</allowDelete>
        <allowEdit>true</allowEdit>
        <allowRead>true</allowRead>
        <viewAllRecords>true</viewAllRecords>
        <modifyAllRecords>true</modifyAllRecords>
        <object>Job_Request__c</object>
    </objectPermissions>
    <fieldPermissions>
        <editable>true</editable>
        <field>Job_Request__c.Salary__c</field>
        <readable>true</readable>
    </fieldPermissions>
    <pageAccesses>
```

```
        <apexPage>Job_Request_Web_Form</apexPage>
        <enabled>true</enabled>
    </pageAccesses>
    <classAccesses>
      <apexClass>Send_Email_Confirmation</apexClass>
      <enabled>true</enabled>
    </classAccesses>
    <tabSettings>
        <tab>Job_Request__c</tab>
        <visibility>Available</visibility>
    </tabSettings>
    <recordTypeVisibilities>
        <recordType>Recruiting.DevManager</recordType>
        <visible>true</visible>
    </recordTypeVisibilities>
</PermissionSet>
```

The following is an example package.xml manifest used to retrieve the PermissionSet metadata for an organization. When you retrieve permission sets, also retrieve the related components with assigned permissions. For example, to retrieve `objectPermissions` and `fieldPermissions` for a custom object, you must also retrieve the CustomObject component.

```
<?xml version="1.0" encoding="UTF-8"?>
<Package xmlns="http://soap.sforce.com/2006/04/metadata">
    <types>
        <members>Job_Request__c</members>
        <name>CustomTab</name>
    </types>
    <types>
        <members>Job_Request__c</members>
        <name>CustomObject</name>
    </types>
    <types>
        <members>JobApps__Recruiting</members>
        <name>CustomApplication</name>
    </types>
    <types>
        <members>Recruiting.DevManager</members>
        <name>RecordType</name>
    </types>
    <types>
        <members>*</members>
        <name>PermissionSet</name>
    </types>
    <version>48.0</version>
</Package>
```

Wildcard Support in the Manifest File

This metadata type supports the wildcard character `*` (asterisk) in the `package.xml` manifest file. For information about using the manifest file, see Deploying and Retrieving Metadata with the Zip File.

PermissionSetGroup

Represents a group of permission *sets* and the permissions within them. Use permission set groups to organize permissions based on job functions or tasks. Then, you can package the groups as needed.

This type extends the Metadata metadata type and inherits its `fullName` field.

Declarative Metadata File Suffix and Directory Location

Permission set groups are stored in the `permissionsetgroups` directory. The file name matches the permission set API name and the extension is `.permissionsetgroup`. For example, a permission set group with the name `Finance_Mgmt_PermSetGroup` is stored in `permissionsetgroups/Finance_Mgmt_PermSetGroup.permissionsetgroup`.

Version

Permission set groups are available in API version 45.0 and later.

Special Access Rules

As of Spring '20 and later, only users who have one of these permissions can access this object:

- View Setup and Configuration
- Manage Session Permission Set Activations
- Manage Users
- Assign Permission Sets
- Manage External Users
- Manage Customer Users

Fields

Field	Field Type	Description
`description`	string	The permission set group description provided by the permission set group creator.
`label`	string	Required. The permission set group label.
`mutingPermissionSets`	string	A permission set containing permissions to disable in the permission set group. Available in API version 46.0 and later.
`permissionSets`	string	A permission set or permission sets included in the permission set group.
`status`	string	Indicates permission set group recalculation status. Valid values are: • `Updated`—The group is current.

Field	Field Type	Description
		• `Outdated`—The group requires recalculation. • `Updating`—The group is in recalculation mode. • `Failed`—The group recalculation failed.

Declarative Metadata Sample Definition

When adding a permission set group, you can do something like this. Individual permissions are included in the permission set referenced, not in the permission set group.

```
<?xml version="1.0" encoding="UTF-8"?>
<PermissionSetGroup xmlns="http://soap.sforce.com/2006/04/metadata">
     <fullName>Finance_Mgmt_PermSetGroup</fullName>
     <description>Finance_Mgmt_PermSetGroup desc</description>
     <label>Finance_Mgmt_PermSetGroup</label>
     <permissionSets>Billing_PS</permissionSets>
</PermissionSetGroup>
```

The permission set `Billing_PS` contains the individual permissions included in `Finance_Mgmt_PermSetGroup`.

```
<?xml version="1.0" encoding="UTF-8"?>
<PermissionSet xmlns="http://soap.sforce.com/2006/04/metadata">
     <fullName>Billing_PS</fullName>
     <description>Billing_PS</description>
     <label>Billing_PS</label>
     <license>Salesforce</license>
     <userPermissions>
          <enabled>true</enabled>
          <name>ViewSetup</name>
     </userPermissions>
     <userPermissions>
          <enabled>true</enabled>
          <name>ViewRoles</name>
     </userPermissions>
     <userPermissions>
          <enabled>true</enabled>
          <name>EditBillingInfo</name>
    </userPermissions>
</PermissionSet>
```

This example `package.xml` manifest retrieves the PermissionSetGroup metadata for an org. When you retrieve permission set groups, also retrieve the related components. For example, to retrieve PermissionSetGroup, you must also retrieve PermissionSet.

```
<?xml version="1.0" encoding="UTF-8"?>
<Package xmlns="http://soap.sforce.com/2006/04/metadata">
     <types>
          <members>Finance_Mgmt_PermSetGroup</members>
          <name>PermissionSetGroup</name>
     </types>
     <types>
          <members>Billing_PS</members>
          <name>PermissionSet</name>
```

```
    </types>
    <version>45.0</version>
</Package>
```

PlatformCachePartition

Represents a partition in the Platform Cache. This type extends the Metadata metadata type and inherits its `fullName` field.

File Suffix and Directory Location

PlatformCachePartition components have the suffix `.cachePartition` and are stored in the `cachePartitions` folder.

Version

PlatformCachePartition components are available in API version 35.0 and later.

Special Access Rules

The “Author Apex” permission is required to deploy and retrieve PlatformCachePartition components.

Fields

Field Name	Field Type	Description
`description`	string	Describes the cache partition.
`isDefaultPartition`	boolean	Required. Indicates whether this cache partition is the default partition in your organization (`true`) or not (`false`).
`masterLabel`	string	Required. The label of the cache partition that appears in the Salesforce user interface.
`platformCachePartitionTypes`	PlatformCachePartitionType[]	An array of cache types that the partition can store.

PlatformCachePartitionType

Contains information about a partition type, including its minimum and allocated capacity.

Field Name	Field Type	Description
`allocatedCapacity`	int	Required. The total storage capacity, in MB, that is allocated for the cache type, including free, purchased, and trial cache. Purchased capacity includes organization-wide cache, which can be used in any partition, and namespace-specific cache, which can be used only in partitions associated with a namespace.
`allocatedPurchasedCapacity`	int	Required. The amount of namespace-specific purchased storage capacity, in MB, that is allocated for the cache type.

Field Name	Field Type	Description
`allocatedTrialCapacity`	int	Required. The amount of trial cache space, in MB, that is allocated for the cache type.
`cacheType`	PlatformCacheType (enumeration of type string)	The type of cache. Valid values are: • `Session`—Session cache • `Organization`—Org cache

Declarative Metadata Sample Definition

The following is an example of a PlatformCachePartition component.

```
<?xml version="1.0" encoding="UTF-8"?>
<PlatformCachePartition xmlns="http://soap.sforce.com/2006/04/metadata">
    <description>Custom partition and marked as default.</description>
    <isDefaultPartition>true</isDefaultPartition>
    <masterLabel>myPartition</masterLabel>
    <platformCachePartitionTypes>
        <allocatedCapacity>10</allocatedCapacity>
        <allocatedPurchasedCapacity>5</allocatedPurchasedCapacity>
        <cacheType>Session</cacheType>
    </platformCachePartitionTypes>
    <platformCachePartitionTypes>
        <allocatedCapacity>5</allocatedCapacity>
        <allocatedPurchasedCapacity>5</allocatedPurchasedCapacity>
        <cacheType>Organization</cacheType>
    </platformCachePartitionTypes>
</PlatformCachePartition>
```

The following is an example `package.xml` that references the previous definition.

```
<?xml version="1.0" encoding="UTF-8"?>
<Package xmlns="http://soap.sforce.com/2006/04/metadata">
    <types>
        <members>myPartition</members>
        <name>PlatformCachePartition</name>
    </types>
    <version>48.0</version>
</Package>
```

If a namespace is defined in your organization, add the namespace prefix to your partition name. For example:

```
<members>Namespace.myPartition</members>
```

To retrieve all cache partitions from your organization, use the wildcard character (*) as follows.

```
<?xml version="1.0" encoding="UTF-8"?>
<Package xmlns="http://soap.sforce.com/2006/04/metadata">
    <types>
        <members>*</members>
        <name>PlatformCachePartition</name>
    </types>
    <version>48.0</version>
</Package>
```

Wildcard Support in the Manifest File

This metadata type supports the wildcard character `*` (asterisk) in the `package.xml` manifest file. For information about using the manifest file, see Deploying and Retrieving Metadata with the Zip File.

PlatformEventChannel

Represents a Change Data Capture standard or custom channel that you can subscribe to for receiving notifications of Salesforce record changes. The standard channel corresponds to the entity selection in the Change Data Capture page in Setup. A custom channel is a channel that you define using this metadata type. Starting in API version 47.0, the channel doesn't contain the selected entities, which are represented each by PlatformEventChannelMember. In API version 46.0 and earlier, the channel includes the selected entities. Change Data Capture sends notifications for created, updated, deleted, and undeleted records only for the selected entities. This type extends the Metadata on page 652 metadata type and inherits its `fullName` field.

File Suffix and Directory Location

PlatformEventChannel components have the suffix `.platformEventChannel` and are stored in the `platformEventChannels` folder.

Version

PlatformEventChannel components are available in API version 45.0 and later.

Special Access Rules

You must have the Customize Application permission to deploy and retrieve this type.

Fields

Field Name	Field Type	Description
`channelMembers`	PlatformEventChannel SelectedEntity[]	Removed. A list of event names of entities, including standard and custom objects, selected for Change Data Capture notifications. Note: This field is removed in API version 47.0 and later and is available only in API versions 45.0 and 46.0. In API version 47.0 and later, the channel members are each defined in a PlatformEventChannelMember component.
`channelType`	PlatformEventChannel Type (enumeration of type string)	Required. The channel type. Valid values are: • `data`—Change Data Capture channel corresponding to the selected entities. • `event`—Reserved for future use.

Field Name	Field Type	Description
`label`	string	Required. The channel label.

PlatformEventChannelSelectedEntity

Note: This field type is removed in API version 47.0 and later and is available only in API versions 45.0 and 46.0.

Field Name	Field Type	Description
`selectedEntity`	string	Required. The event name of an entity selected for Change Data Capture notifications. For example, for the Account standard object, the name is `AccountChangeEvent`, or for a custom object MyObject__c, the name is `MyObject__ChangeEvent`.

Usage

The createMetadata() and deleteMetadata() calls aren't supported with the PlatformEventChannel metadata type.

In API version 47.0 and later, you can't deploy or retrieve the ChangeEvents standard channel.

You can't delete the ChangeEvents standard channel with `destructiveChanges.xml`, but you can delete channel members using the PlatformEventChannelMember type with `destructiveChanges.xml`.

Declarative Metadata Sample Definition for a Custom Channel

The PlatformEventChannel component contains the label of the custom channel and the channel type.

```
<?xml version="1.0" encoding="UTF-8"?>
<PlatformEventChannel xmlns="http://soap.sforce.com/2006/04/metadata">
    <channelType>data</channelType>
    <label>Custom Channel for Sales Events</label>
</PlatformEventChannel>
```

This `package.xml` references the previous definition. The custom channel name is `SalesEvents__chn`.

```
<?xml version="1.0" encoding="UTF-8"?>
<Package xmlns="http://soap.sforce.com/2006/04/metadata">
    <types>
        <members>SalesEvents__chn</members>
        <name>PlatformEventChannel</name>
    </types>
    <version>48.0</version>
</Package>
```

Wildcard Support in the Manifest File

To deploy or retrieve all custom channels, specify the wildcard character `*` (asterisk) in the `<members>` field.

```
<?xml version="1.0" encoding="UTF-8"?>
<Package xmlns="http://soap.sforce.com/2006/04/metadata">
    <types>
        <members>*</members>
        <name>PlatformEventChannel</name>
    </types>
    <version>48.0</version>
</Package>
```

Upgrading to Version 47.0 or Later From an Earlier Version

The `channelMembers` field of the PlatformEventChannel type is removed in API version 47.0 and later. As a result, PlatformEventChannel components created in prior versions can’t be deployed using a later API version but you can deploy them in the same API version they were created with.

To deploy a custom channel component using API version 47.0 and later, upgrade the PlatformEventChannel definition by removing the `<channelMembers>` fields. For the ChangeEvents standard channel, it can’t be deployed or retrieved, so delete the PlatformEventChannel definition file.

For example, if you had custom channel called SalesEvents__chn, this could be your custom channel definition in API version 46.0.

```
<?xml version="1.0" encoding="UTF-8"?>
<PlatformEventChannel xmlns="http://soap.sforce.com/2006/04/metadata">
    <channelMembers>
        <selectedEntity>AccountChangeEvent</selectedEntity>
    </channelMembers>
    <channelMembers>
        <selectedEntity>ContactChangeEvent</selectedEntity>
    </channelMembers>
    <channelType>data</channelType>
    <label>Sales Events</label>
</PlatformEventChannel>
```

To upgrade to version 47.0 or later, you would replace the custom channel definition with this definition, which doesn’t contain any channel members.

```
<?xml version="1.0" encoding="UTF-8"?>
<PlatformEventChannel xmlns="http://soap.sforce.com/2006/04/metadata">
    <channelType>data</channelType>
    <label>SalesEvents__chn</label>
</PlatformEventChannel>
```

For each channel member that is part of either a custom or the standard ChangeEvents channel, add a PlatformEventChannelMember metadata component. Also, in the `package.xml` file, reference both the PlatformEventChannel and PlatformEventChannelMember components.

For example, this PlatformEventChannelMember component is for the AccountChangeEvent member.

```
<?xml version="1.0" encoding="UTF-8"?>
<PlatformEventChannelMember xmlns="http://soap.sforce.com/2006/04/metadata">
    <eventChannel>SalesEvents__chn</eventChannel>
    <selectedEntity>AccountChangeEvent</selectedEntity>
</PlatformEventChannelMember>
```

For more information, see PlatformEventChannelMember.

PlatformEventChannelMember

Represents an entity selected for Change Data Capture notifications on a standard or custom channel. This type extends the Metadata on page 652 metadata type and inherits its `fullName` field.

File Suffix and Directory Location

PlatformEventChannelMember components have the suffix `.platformEventChannelMember` and are stored in the `platformEventChannelMembers` folder.

Version

PlatformEventChannelMember components are available in API version 47.0 and later.

Special Access Rules

You must have the Customize Application permission to deploy and retrieve this type.

Fields

Field Name	Field Type	Description
`eventChannel`	string	Required. The name of a channel. For the standard channel, the name is `ChangeEvents`. For a custom channel, the name is in this format: ***MyChannel***`__chn`.
`selectedEntity`	string	Required. The change event name of an entity selected for Change Data Capture notifications. For example, for the Account standard object, the name is `AccountChangeEvent`, or for a custom object MyObject__c, the name is `MyObject__ChangeEvent`.

Usage

The createMetadata() and deleteMetadata() calls aren't supported with the PlatformEventChannelMember metadata type.

To delete a channel member from a channel, deploy `destructiveChanges.xml` for this type and specify the full name of the member.

Declarative Metadata Sample Definition

This PlatformEventChannelMember component represents the selection of the Lead change event as part of the Change Data Capture selections (the standard `ChangeEvents` channel).

```
<?xml version="1.0" encoding="UTF-8"?>
<PlatformEventChannelMember xmlns="http://soap.sforce.com/2006/04/metadata">
    <eventChannel>ChangeEvents</eventChannel>
    <selectedEntity>LeadChangeEvent</selectedEntity>
</PlatformEventChannelMember>
```

Note: The file name of the example component is `ChangeEvents_LeadChangeEvent.platformEventChannelMember`. The file name, without the extension, corresponds to the component full name (`ChangeEvents_LeadChangeEvent`).

If the channel has more than one selected entity, each entity is represented separately by a PlatformEventChannelMember component. For example, this component is a second member of the standard ChangeEvents channel and represents the Contact change event.

```
<?xml version="1.0" encoding="UTF-8"?>
<PlatformEventChannelMember xmlns="http://soap.sforce.com/2006/04/metadata">
    <eventChannel>ChangeEvents</eventChannel>
    <selectedEntity>ContactChangeEvent</selectedEntity>
</PlatformEventChannelMember>
```

This example is a selected entity on the `SalesEvents__chn` custom channel.

```
<?xml version="1.0" encoding="UTF-8"?>
<PlatformEventChannelMember xmlns="http://soap.sforce.com/2006/04/metadata">
    <eventChannel>SalesEvents__chn</eventChannel>
    <selectedEntity>ContactChangeEvent</selectedEntity>
</PlatformEventChannelMember>
```

Underscores in Channel Member Full Names

Two consecutive underscores in full names designate either a component name suffix or a namespace prefix. In all other cases, two consecutive underscores aren't supported in full names. If your channel member name contains a custom channel name to make it unique, ensure to replace the double underscores in the name with one underscore. For example, the member name would be `SalesEvents_chn_AccountChangeEvent` and not `SalesEvents__chn_AccountChangeEvent`.

Referencing Channel Members and Channels in `Package.xml`

This manifest file references the example definitions on the `ChangeEvents` standard channel. It lists each member in the `<members>` field of `PlatformEventChannelMember`. The `<members>` field contains the channel member full name in this format: ***`ChannelName_EventName`***.

```
<?xml version="1.0" encoding="UTF-8"?>
<Package xmlns="http://soap.sforce.com/2006/04/metadata">
    <types>
        <members>ChangeEvents_LeadChangeEvent</members>
        <members>ChangeEvents_ContactChangeEvent</members>
        <name>PlatformEventChannelMember</name>
    </types>
    <version>48.0</version>
</Package>
```

This manifest file references members of the `SalesEvents__chn` custom channel.

```
<?xml version="1.0" encoding="UTF-8"?>
<Package xmlns="http://soap.sforce.com/2006/04/metadata">
    <types>
        <members>SalesEvents_chn_AccountChangeEvent</members>
        <members>SalesEvents_chn_ContactChangeEvent</members>
        <members>SalesEvents_chn_MyCustomObj_ChangeEvent</members>
        <name>PlatformEventChannelMember</name>
    </types>
    <version>47.0</version>
</Package>
```

To retrieve a custom channel and channel members, you can reference them in the same `package.xml` file, as this example shows.

```
<?xml version="1.0" encoding="UTF-8"?>
<Package xmlns="http://soap.sforce.com/2006/04/metadata">
    <types>
        <members>SalesEvents__chn</members>
        <name>PlatformEventChannel</name>
    </types>
    <types>
        <members>SalesEvents_chn_AccountChangeEvent</members>
        <members>SalesEvents_chn_ContactChangeEvent</members>
        <members>SalesEvents_chn_MyCustomObj_ChangeEvent</members>
        <name>PlatformEventChannelMember</name>
    </types>
    <version>47.0</version>
</Package>
```

Wildcard Support in the Manifest File

This metadata type supports the wildcard character `*` (asterisk) in the `package.xml` manifest file. For information about using the manifest file, see Deploying and Retrieving Metadata with the Zip File.

Portal

The Portal metadata type represents a partner portal or Customer Portal.

It extends Metadata and inherits its `fullName` field. To use this metadata type, you must have a partner portal or Customer Portal enabled for your organization. For more information, see “Partner Portal Overview” and “Enabling Your Customer Portal” in the Salesforce online help.

Declarative Metadata File Suffix and Directory Location

Lightning Platform Portal components are stored in the `portals` directory of the corresponding package directory. The file name matches the portal name, and the extension is `.portal`.

Version

Lightning Platform Portal components are available in API version 15.0 and later.

Fields

Field	Field Type	Description
`active`	boolean	Required. Denotes whether this portal is active.
`admin`	string	The full name of the user designated to administer the portal.
`defaultLanguage`	string	The default language for HTML messages for the portal. Use the abbreviation for the language, for example, en_US for United States English.
`description`	string	The portal description.
`emailSenderAddress`	string	Required. The email address used when sending emails using templates configured from the portal (for example, for resetting the password).
`emailSenderName`	string	Required. The name to display when sending emails using templates configured from the portal (for example, for resetting the password).
`enableSelfCloseCase`	boolean	For the Customer Portal, allows portal users to close their own cases.
`footerDocument`	string	The file to be used as the footer for this portal.
`forgotPassTemplate`	string	The email template to use when a user clicks the **Forgot Password** link. Lightning email templates aren't packageable. We recommend using a Classic email template.
`fullName`	string	Required. The name of the portal. Inherited from Metadata, this field is not defined in the WSDL for this metadata type. It must be specified when creating, updating, or deleting. See `createMetadata()` to see an example of this field specified for a call.
`headerDocument`	string	The file to be used as the header for this portal.
`isSelfRegistrationActivated`	boolean	Determines whether self-registration is active or not for this portal.
`loginHeaderDocument`	string	The file to be used as the header for this portal's login page.
`logoDocument`	string	The file to be used as the logo for this portal.
`logoutUrl`	string	The URL that the user should be redirected to on logout.

Field	Field Type	Description
`newCommentTemplate`	string	The email template to be used for auto-notifications on new case comments.
`newPassTemplate`	string	The email template to be used for auto-notifications on password reset. Lightning email templates aren't packageable. We recommend using a Classic email template.
`newUserTemplate`	string	The email template to be used for auto-notifications on new user creation. Lightning email templates aren't packageable. We recommend using a Classic email template.
`ownerNotifyTemplate`	string	The email template to be used for auto-notifications on owner change. Lightning email templates aren't packageable. We recommend using a Classic email template.
`selfRegNewUserUrl`	string	The URL of the self-registration page.
`selfRegUserDefaultProfile`	string	The default profile for self-registered users.
`selfRegUserDefaultRole`	PortalRoles (enumeration of type string)	The default role for self-registered users. The valid values are: • Executive • Manager • User • PersonAccount
`selfRegUserTemplate`	string	The email template to be used for auto-notifications on self-registration. Lightning email templates aren't packageable. We recommend using a Classic email template.
`showActionConfirmation`	boolean	Determines whether or not confirmation messages are displayed for actions in the portal.
`stylesheetDocument`	string	The Document object to be used as the CSS stylesheet for this portal.
`type`	PortalType (enumeration of type string)	Required. The type for this portal. The valid values are: • CustomerSuccess • Partner

Declarative Metadata Sample Definition

A sample XML definition of a portal is shown below.

```
<?xml version="1.0" encoding="UTF-8"?>
<Portal xmlns="http://soap.sforce.com/2006/04/metadata">
    <active>true</active>
    <description>Customer Portal</description>
    <emailSenderName>rguest@albany.com</emailSenderName>
    <enableSelfCloseCase>false</enableSelfCloseCase>
    <forgotPassTemplate>unfiled$public/ChangePwdEmail</forgotPassTemplate>
    <isSelfRegistrationActivated>false</isSelfRegistrationActivated>
    <newPassTemplate>unfiled$public/ChangePwdEmail</newPassTemplate>
    <newUserTemplate>unfiled$public/NewUserEmail</newUserTemplate>
    <selfRegUserTemplate>unfiled$public/SelfRegUserEmail</selfRegUserTemplate>
    <showActionConfirmation>false</showActionConfirmation>
    <type>CustomerSuccess</type>
</Portal>
```

Wildcard Support in the Manifest File

This metadata type supports the wildcard character `*` (asterisk) in the `package.xml` manifest file. For information about using the manifest file, see Deploying and Retrieving Metadata with the Zip File.

SEE ALSO:

CustomSite

PostTemplate

Represents the metadata associated with an approval post template for Approvals in Chatter. With approval post templates, you can customize the information included in approval request posts that appear in Chatter feeds. This type extends the Metadata metadata type and inherits its `fullName` field.

Note: Review Chatter Post Templates for Approval Requests in the Salesforce Help before you create a post template.

File Suffix and Directory Location

PostTemplate components have the suffix `.postTemplate` and are stored in the `postTemplates` folder.

Version

PostTemplate components are available in API version 29.0 and later.

Fields

Field Name	Field Type	Description
`default`	boolean	Required. Specifies whether this is the default post template for the given object. When set to `true`, this post template is used by approval processes that are associated with the same object and don't specify a post template. When an object has no default post template, each of its approval processes uses the system default post template, unless the approval process specifies its own post template.
`description`	string	Optional description of the post template.
`fields`	string[]	Required. An array of up to four fields to include in approval request posts. If the approval object is a detail object in a master-detail relationship, `Owner` isn't available for approval page layouts or approval post templates.
`label`	string	Required. Name of the post template. This non-unique label is different from the unique name of the post template.

Declarative Metadata Sample Definition

The following is an example of a PostTemplate component:

```
<PostTemplate xmlns="http://soap.sforce.com/2006/04/metadata">
    <default>false</default>
    <fields>NumberOfEmployees</fields>
    <fields>NumberofLocations__c</fields>
    <fields>PartnerAccount</fields>
    <fields>LeadCustomFieldNumber__c</fields>
    <label>My Lead Post Template</label>
</PostTemplate>
```

The following is an example package manifest that references the previous PostTemplate component.

```
<?xml version="1.0" encoding="UTF-8"?>
<Package xmlns="http://soap.sforce.com/2006/04/metadata">
    <types>
        <members>Lead.leadtemplate</members>
        <name>PostTemplate</name>
    </types>
    <version>29.0</version>
</Package>
```

Wildcard Support in the Manifest File

This metadata type supports the wildcard character `*` (asterisk) in the `package.xml` manifest file. For information about using the manifest file, see Deploying and Retrieving Metadata with the Zip File.

PresenceDeclineReason

Represents an Omni-Channel decline reason that agents can select when declining work requests. This type extends the Metadata metadata type and inherits its `fullName` field.

File Suffix and Directory Location

PresenceDeclineReason components have the suffix `.presenceDeclineReason` and are stored in the `presenceDeclineReasons` folder.

Version

PresenceDeclineReason components are available in API version 44.0 and later.

Special Access Rules

This type is available only if Omni-Channel is enabled in your org.

Fields

Field Name	Field Type	Description
`label`	string	The master label for the decline reason.

Declarative Metadata Sample Definition

The following is an example of a PresenceDeclineReason component.

```
<?xml version="1.0" encoding="UTF-8"?>
<PresenceDeclineReason xmlns="http://soap.sforce.com/2006/04/metadata">
    <label>Incorrect queue</label>
</PresenceDeclineReason>
```

The following is an example `package.xml` that references the previous definition.

```
<?xml version="1.0" encoding="UTF-8"?>
<Package xmlns="http://soap.sforce.com/2006/04/metadata">
    <types>
        <members>*</members>
        <name>PresenceDeclineReason</name>
    </types>
    <version>44.0</version>
</Package>
```

Wildcard Support in the Manifest File

This metadata type supports the wildcard character `*` (asterisk) in the `package.xml` manifest file. For information about using the manifest file, see Deploying and Retrieving Metadata with the Zip File.

PresenceUserConfig

Represents a configuration that determines a presence user's settings. This type extends the Metadata metadata type and inherits its `fullName` field.

File Suffix and Directory Location

PresenceUserConfig components have the suffix `.presenceUserConfig` and are stored in the `presenceUserConfigs` folder.

Version

PresenceUserConfig components are available in API version 44.0 and later.

Special Access Rules

This type is available only if Omni-Channel is enabled in your org.

Fields

Field Name	Field Type	Description
`assignments`	PresenceConfigAssignments	Specifies how presence configurations are assigned to Omni-Channel users. Presence configurations can be assigned to sets of users or to sets of profiles.
`capacity`	int	Required. The maximum number of work assignments that can be pushed to an agent at a time.
`declineReasons`	string	Specifies the list of decline reasons that an agent can select when they decline a work.
`enableAutoAccept`	boolean	Indicates whether work items that are routed to agents are automatically accepted (`true`) or not (`false`). Available only if `enableDecline` is set to `false`.
`enableDecline`	boolean	Indicates whether agents can decline work items that are routed to them (`true`) or not (`false`). Available only if `enableAutoAccept` is set to `false`.
`enableDeclineReason`	boolean	Indicates whether agents can select a reason for declining work requests (`true`) or not (`false`). This can be selected only if decline reasons are enabled.
`enableDisconnectSound`	boolean	Indicates whether a sound is played when agents are disconnected from Omni-Channel (`true`) or not (`false`).
`enableRequestSound`	boolean	Indicates whether a sound plays with incoming work requests (`true`) or not (`false`). Set to `true` by default.

Field Name	Field Type	Description
`label`	string	The label of the presence configuration.
`presenceStatusOnDecline`	string	The presence status that's automatically assigned to the agent when the agent declines a work item. Available only if `enableDecline` is set to `true`.
`presenceStatusOnPushTimeout`	string	The presence status that's automatically assigned to the agent when the agent doesn't respond to a work item before push timeout occurs.

PresenceConfigAssignments

Represents the assignments of an org's profiles and users to a Presence configuration.

Field Name	Field Type	Description
`profiles`	PresenceConfigProfileAssignments	Specifies the profiles that are associated with a specific presence configuration.
`users`	PresenceConfigUserAssignments	Specifies the users that are associated with a specific presence configuration.

PresenceConfigProfileAssignments

Represents the profiles associated with a specific presence configuration.

Field Name	Field Type	Description
`profile`	string	Specifies the name of the profile associated with a specific presence configuration.

PresenceConfigUserAssignments

Represents the users associated with a specific presence configuration.

Field Name	Field Type	Description
`user`	string	Specifies the username of the user associated with a specific presence configuration.

Declarative Metadata Sample Definition

The following is an example of a PresenceUserConfig component.

```
<?xml version="1.0" encoding="UTF-8"?>
<PresenceUserConfig xmlns="http://soap.sforce.com/2006/04/metadata">
    <assignments>
        <profiles>
            <profile>standard</profile>
        </profiles>
        <users>
            <user>jdoe@example.com</user>
        </users>
    </assignments>
    <capacity>5</capacity>
    <declineReasons>Incorrect_queue</declineReasons>
    <enableAutoAccept>false</enableAutoAccept>
    <enableDecline>true</enableDecline>
    <enableDeclineReason>true</enableDeclineReason>
    <enableDisconnectSound>true</enableDisconnectSound>
    <enableRequestSound>true</enableRequestSound>
    <label>My presence configuration</label>
    <presenceStatusOnDecline>Away</presenceStatusOnDecline>
    <presenceStatusOnPushTimeout>Break</presenceStatusOnPushTimeout>
</PresenceUserConfig>
```

The following is an example `package.xml` that references the previous definition.

```
<?xml version="1.0" encoding="UTF-8"?>
<Package xmlns="http://soap.sforce.com/2006/04/metadata">
    <types>
        <members>*</members>
        <name>PresenceUserConfig</name>
    </types>
    <version>44.0</version>
</Package>
```

Wildcard Support in the Manifest File

This metadata type supports the wildcard character `*` (asterisk) in the `package.xml` manifest file. For information about using the manifest file, see Deploying and Retrieving Metadata with the Zip File.

Profile

Represents a user profile. A profile defines a user's permission to perform different functions within Salesforce. This type extends the Metadata metadata type and inherits its `fullName` field.

In API version 29.0 and later, you can retrieve and deploy access settings for the following managed components in profiles and permission sets:

- Apex classes
- Apps
- Custom field permissions
- Custom object permissions
- Custom tab settings
- External data sources
- Record types
- Visualforce pages

For more information, see Managed Component Access in Sample `package.xml` Manifest Files on page 23.

Declarative Metadata File Suffix and Directory Location

The file suffix is `.profile`. There is one file for each profile, stored in the `profiles` folder in the corresponding package directory.

Version

Profiles are available in API version 10.0 and later.

Fields

The content of a profile returned by Metadata API depends on the content requested in the `RetrieveRequest` message. For example, profiles only include field-level security for fields included in custom objects returned in the same `RetrieveRequest` as the profiles.

Important: We designed Profile metadata deployment to overlay the existing Profile settings in a target org. For example, if you disable permissions for a profile, the newly disabled permission information isn't exported. To force all Profile changes to deploy through metadata, including permission disablement, add code that explicitly indicates disabled permissions. For example, add the following code to the Profile metadata `.xml` file before deploying into a target org: `<value>false</value>`.

Note: As of API version 38.0, you can change field permissions to make a field editable using the Metadata API for fields that you can't change through the user interface. For example, you can deploy `Asset.ProductCode` as an editable field even though you can't through the user interface.

The profile definition contains the following fields:

Field Name	Field Type	Description
`applicationVisibilities`	ProfileApplicationVisibility[]	Indicates which apps are visible to users assigned to this profile. In API version 29.0 and earlier, this field supports custom apps only. In API version 30.0 and later, this field supports both standard and custom apps.
`categoryGroupVisibilities`	ProfileCategoryGroupVisibility[]	Indicates which data category groups are visible to users assigned to this profile. Available in API version 41.0 and later.
`classAccesses`	ProfileApexClassAccess[]	Indicates which top-level Apex classes have methods that users assigned to this profile can execute.
`custom`	boolean	Indicates whether the profile is a custom (`true`) or standard (`false`) profile. Available in API version 30.0 and later.
`customMetadataTypeAccesses`	ProfileCustomMetadataTypeAccess[]	Indicates the custom metadata types that are read-accessible to a user assigned to this profile. Available in API version 47.0 and later.
`customPermissions`	ProfileCustomPermissions[]	Indicates which custom permissions are available to users assigned to this profile. Available in API version 31.0 and later.

Field Name	Field Type	Description
customSettingAccesses	ProfileCustomSettingAccesses[]	Indicates the custom settings that are read-accessible to a user assigned to this profile. Available in API version 47.0 and later.
description	string	The profile description. Limit: 255 characters. Available in API version 30.0 and later.
externalDataSourceAccesses	ProfileExternalDataSourceAccess[]	Indicates which data sources with identity type of Per User are available to users assigned to this profile. Available in API version 27.0 and later.
fieldLevelSecurities	ProfileFieldLevelSecurity[]	Indicates which fields are visible to a user assigned to this profile, and the kind of access available (editable or hidden). This field is available in API version 22.0 and earlier.
fieldPermissions	ProfileFieldLevelSecurity[]	Indicates which fields are visible to a user assigned to this profile, and the kind of access available (editable or readable). This field is available in API version 23.0 and later.
fullName	string	The name can only contain characters, letters, and the underscore (_) character, must start with a letter, and cannot end with an underscore or contain two consecutive underscore characters. Inherited from the Metadata component, this field is not defined in the WSDL for this component. It must be specified when creating, updating, or deleting. See create() to see an example of this field specified for a call.
layoutAssignments	ProfileLayoutAssignments[]	Indicates which layout to use for this profile.
loginHours	ProfileLoginHours[]	Indicates the hours within which a user with this profile can log in. If not specified, the profile doesn't restrict a user's login hours. This field is available in API version 25.0 and later.
loginIpRanges	ProfileLoginIpRange[]	The list of IP address ranges from which users with a particular profile can log in. This field is available in API version 17.0 and later.
objectPermissions	ProfileObjectPermissions[]	Indicates which objects are accessible to a user assigned to this profile, and the kind of access available (create, read, edit, delete, and so on). In API version 28.0 and later, this field is only retrieved when allowRead is true.
pageAccesses	ProfileApexPageAccess[]	Indicates which Visualforce pages that users assigned to this profile can execute.

Field Name	Field Type	Description
`profileActionOverrides`	ProfileActionOverride[]	A list of the Lightning Experience Home page action overrides that are assigned to this profile. When a user logs in with a profile, a matching ProfileActionOverride assignment takes precedence over existing overrides for the Home tab specified in ActionOverride. This field is available in API versions 37.0 to 44.0.
`recordTypeVisibilities`	ProfileRecordTypeVisibility[]	Indicates the visibility of record types for users assigned to this profile. In API version 29.0 and later, this field is not retrieved or deployed for inactive record types.
`tabVisibilities`	ProfileTabVisibility[]	Indicates which record types are visible to a user assigned to this profile, and therefore which tabs within an app are visible.
`userLicense`	string	The `User License` for the profile. A user license determines the baseline of features that the user can access. Every user must have exactly one user license. This field is available in API version 17.0 and later.
`userPermissions`	ProfileUserPermission[]	Specifies a user permission (such as "API Enabled") and whether it's enabled for this profile. This field retrieves only enabled user permissions. Available in API version 29.0 and later.

ProfileActionOverride

ProfileActionOverride represents a user profile-based override of an ActionOverride on a standard Home tab in Lightning Experience.

Note:

- ProfileActionOverride can be defined only on Profile for API version 39.0 to 44.0. In API version 45.0 and later, ProfileActionOverride must be defined for CustomApplication instead. Beginning with API version 45.0, Home page assignments related to user profile must also have a corresponding app assignment because more granular Home page assignments are supported. As a result, ProfileActionOverride is defined for CustomApplication rather than Profile.
- ProfileActionOverride settings aren't retrieved in the `.profile` file unless a Lightning page is referenced in the `package.xml` file.

Field Name	Field Type	Description
`actionName`	string	Required. The possible values are the same as the actions you can override: • `accept` • `clone` • `delete` • `edit`

Field Name	Field Type	Description
		- `list` - `new` - `tab` - `view`
`content`	string	Set this field if `type` is set to `flexipage`, `lightningcomponent`, `scontrol`, or `visualforce`. It refers to the name of the Lightning page, Lightning component, s-control, or Visualforce page to use as the override. To reference installed components, use this format: ***`Component_namespace__Component_name`***.
`formFactor`	FormFactor (enumeration of type string)	The size of the page being overridden. The `Large` value represents the Lightning Experience desktop environment and is valid only for the `flexipage` and `lightningcomponent` types. The `Small` value represents the Salesforce app on a phone or tablet. The `Medium` value is reserved for future use. The `null` value (which is the same as specifying no value) represents Salesforce Classic.
`pageOrSobjectType`	string	The name of the sObject type being overridden. Valid values are `standard` and `custom`. This value must be `standard-home` when actionName is `tab`.
`recordType`	string	The record type assigned to the ProfileActionOverride. If the `PageOrSobjectType` is `standard-home`, this field is null.
`type`	ActionOverrideType (enumeration of type string)	Required. Represents the type of action override. Valid values are described in ActionOverrideType.

ProfileApplicationVisibility

ProfileApplicationVisibility determines whether an app is visible to a user assigned to this profile.

Field Name	Field Type	Description
`application`	string	Required. The name of the app.
`default`	boolean	Required. Indicates whether the app is the default app (`true`) or not (`false`). Only one app per profile can be set to `true`.
`visible`	boolean	Required. Indicates whether this app is visible to users assigned to this profile (`true`) or not (`false`).

ProfileCategoryGroupVisibility

ProfileCategoryGroupVisibility determines whether a data category group is visible to a user assigned to this profile. Available in API version 41.0 and later.

Field Name	Field Type	Description
`dataCategories`	string[]	Array of one or more data category names.
`dataCategoryGroup`	string	Required. The name of the data category group.
`visibility`	CategoryGroupVisibility (enumeration of type string)	Required. Indicates the visibility of the data category. Valid values are: • `ALL` • `CUSTOM` • `NONE`

ProfileCustomMetadataTypeAccess

ProfileCustomMetadataTypeAccess represents the custom metadata type access for users assigned to a profile. Available in API version 47.0 and later.

Field	Field Type	Description
`enabled`	boolean	Required. Indicates whether the records for this custom metadata type are readable (`true`) or not (`false`).
`name`	string	Required. The custom metadata type name.

ProfileApexClassAccess

ProfileApexClassAccess determines which top-level Apex classes have methods that users assigned to this profile can execute.

Field Name	Field Type	Description
`apexClass`	string	Required. The Apex class name.
`enabled`	boolean	Required. Indicates whether users assigned to this profile can execute methods in the top-level class (`true`) or not (`false`).

ProfileCustomPermissions

ProfileCustomPermissions represents the custom permissions access for users assigned to a profile. Only enabled custom permissions are retrieved.

Field Name	Field Type	Description
enabled	boolean	Required. Indicates whether the custom permission is enabled (true) or not (false).
name	string	Required. The custom permission name.

ProfileCustomSettingAccesses

ProfileCustomSettingAccesses represents the custom setting access for users assigned to a profile. Available in API version 47.0 and later.

Field	Field Type	Description
enabled	boolean	Required. Indicates whether the records for this custom setting are readable (true) or not (false).
name	string	Required. The custom setting name.

ProfileExternalDataSourceAccess

ProfileExternalDataSourceAccess represents the data source access for users with identity type of Per User. Available in API version 27.0 and later.

Field Name	Field Type	Description
enabled	boolean	Required. Indicates whether the data source is enabled (true) or not (false).
externalDataSource	string	The name of the external data source.

ProfileFieldLevelSecurity

ProfileFieldLevelSecurity represents the field level security for users assigned to a profile. In API version 30.0 and later, permissions for required fields can't be retrieved or deployed.

Field Name	Field Type	Description
editable	boolean	Required. Indicates whether this field is editable (true) or not (false). In API version 30.0 and later, when deploying a new custom field, this field is false by default.
field	string	Required. Indicates the name of the field.
hidden	boolean	Indicates whether this field is hidden (true) or not (false). This field is available in API version 22.0 and earlier. For portal profiles, this field is set to true by default in API version 19.0 and later.

Field Name	Field Type	Description
`readable`	boolean	Indicates whether this field is readable (`true`) or not (`false`). This field is available in API version 23.0 and later. It replaces the `hidden` field. In API version 30.0 and later, when deploying a new custom field, this field is `false` by default. For portal profiles, this field is set to `false` by default.

ProfileLayoutAssignments

ProfileLayoutAssignments determines which layout to use for a profile and a given entity.

Field Name	Field Type	Description
`layout`	string	Required. Indicates the layout for this particular entity.
`recordType`	string	This field is optional. If the `recordType` of the record matches a layout assignment rule, it will use the specified layout.

ProfileLoginHours

ProfileLoginHours restricts the days and times within which users with a particular profile can log in.

Field Name	Field Type	Description
`weekday``Start`	string	Specifies the earliest time on that day that a user with this profile can log in. If a start time for a particular day is specified, an end time for that day must be specified as well. Start can't be greater than end for a particular day. • Valid values for `weekday`: `monday`, `tuesday`, `wednesday`, `thursday`, `friday`, `saturday`, or `sunday`. For example, `mondayStart` indicates the beginning of the login period for Monday. • Valid values for Start: the number of minutes since midnight. Must be evenly divisible by 60 (full hours). For example, `300` is 5:00 a.m.
`weekday``End`	string	Specifies the time on that day by which a user with this profile must log out. • Valid values for `weekday`: `monday`, `tuesday`, `wednesday`, `thursday`, `friday`, `saturday`, or `sunday`. For example, `mondayEnd` indicates the close of the login period for Monday. • Valid values for End: the number of minutes since midnight. Must be evenly divisible by 60 (full hours). For example, `1020` is 5:00 p.m.

To delete login hour restrictions from a profile that previously had them, you must explicitly include an empty loginHours tag without any start or end times.

ProfileLoginIpRange

ProfileLoginIpRange IP defines an IP address range from which users with a particular profile can log in.

Field Name	Field Type	Description
`description`	string	Use this field to identify the purpose of the range, such as which part of a network corresponds to this range. This field is available in API version 31.0 and later.
`endAddress`	string	Required. The end IP address for the range.
`startAddress`	string	Required. The start IP address for the range.

ProfileObjectPermissions

ProfileObjectPermissions represents a user's access to objects.

Note: In API version 18.0 and later, these permissions are disabled in new custom objects for any profiles in which “View All Data” or “Modify All Data” is disabled.

Field Name	Field Type	Description
`allowCreate`	boolean	Indicates whether the object referenced by the `object` field can be created by the users assigned to this profile (`true`) or not (`false`). This field is named `revokeCreate` before version 14.0 and the logic is reversed. The field name change and the update from `true` to `false` and the reverse is automatically handled between versions and does not require any manual editing of existing XML component files.
`allowDelete`	boolean	Indicates whether the object referenced by the `object` field can be deleted by the users assigned to this profile (`true`) or not (`false`). This field is named `revokeDelete` before version 14.0 and the logic is reversed. The field name change and the update from `true` to `false` and the reverse is automatically handled between versions and does not require any manual editing of existing XML component files.
`allowEdit`	boolean	Indicates whether the object referenced by the `object` field can be edited by the users assigned to this profile (`true`) or not (`false`). This field is named `revokeEdit` before version 14.0 and the logic is reversed. The field name change and the update from `true` to `false` and the reverse is automatically handled between versions and does not require any manual editing of existing XML component files.

Field Name	Field Type	Description
allowRead	boolean	Indicates whether the object referenced by the object field can be seen by the users assigned to this profile (true) or not (false). This field is named revokeRead before version 14.0 and the logic is reversed. The field name change and the update from true to false and the reverse is automatically handled between versions and does not require any manual editing of existing XML component files.
modifyAllRecords	boolean	Indicates whether the object referenced by the object field can be read, edited, or deleted by the users assigned to this profile (true) or not (false), regardless of the sharing settings for the object. This is equivalent to the "Modify All Data" user permission limited to the individual object level. This is a new field in API version 15.0. Note: This field is not available for all objects. Refer to the profile in the user interface to determine which objects currently support these permissions. Profiles with "Modify All Data" ignore modifyAllRecords entries in Metadata API and don't return an error if "Modify All Data" is enabled on the profile.
object	string	Required. The name of the object whose permissions are altered by this profile, for example, MyCustomObject__c.
viewAllRecords	boolean	Indicates whether the object referenced by the object field can be read by the users assigned to this profile (true) or not (false), regardless of the sharing settings for the object. This includes private records (records with no parent object). This is equivalent to the "View All Data" user permission limited to the individual object level. This is a new field in API version 15.0. Note: This field is not available for all objects. Refer to the profile in the user interface to determine which objects currently support these permissions. Profiles with "View All Data" ignore viewAllRecords entries in the Metadata API and don't return an error if "View All Data" is enabled on the profile.

ProfileApexPageAccess

ProfileApexPageAccess determines which Visualforce pages that users assigned to this profile can execute.

Field Name	Field Type	Description
apexPage	string	Required. The Visualforce page name.
enabled	boolean	Required. Indicates whether users assigned to this profile can execute the Visualforce page (true) or not (false).

ProfileRecordTypeVisibility

ProfileRecordTypeVisibility represents the visibility of record types for this profile. Record types let you offer different business processes, picklist values, and page layouts to different users.

Field Name	Field Type	Description
`default`	boolean	Required. Indicates whether the record type is the default for this pair of profile and object (`true`) or not (`false`). Only one default is allowed per object.
`personAccountDefault`	boolean	Indicates whether the record type is the default person account record type for this pair of profile and object (`true`) or not (`false`). Only one person account record type default is allowed per object. This field is only relevant for record types for account or contact objects. For more information about person accounts, see “Person Accounts” in the Salesforce online help. Person accounts are not enabled by default in Salesforce. To request person accounts, contact Salesforce.
`recordType`	string	Required. The record type name, for example `Account.MyRecordType`.
`visible`	boolean	Required. Indicates whether this record type is visible to users assigned to this profile (`true`) or not (`false`).

ProfileTabVisibility

ProfileTabVisibility represents the visibility of tabs for this profile. For version 17.0 and later, ProfileTabVisibility supports visibility of tabs for standard objects. The manifest file must include the standard object corresponding to a standard tab to retrieve the tab visibility in a profile.

Field Name	Field Type	Description
`tab`	string	Required. The name of the tab.
`visibility`	TabVisibility (enumeration of type string)	Required. Indicates the visibility of the tab. Valid values are: • `DefaultOff`—The tab is available on the All Tabs page. Users can individually customize their display to make the tab visible in any app. • `DefaultOn`—The tab is available on the All Tabs page and appears in the visible tabs for its associated app. Users can individually customize their display to hide the tab or make it visible in other apps. • `Hidden`—The tab isn’t available on the All Tabs page or visible in any apps. Note: In version 36.0 and earlier, `Hidden` is returned only if `visibility` was set using the API. If it was set to `Hidden` from the profile in Salesforce, the API doesn’t return a visibility

Field Name	Field Type	Description
		value. For version 37.0 and later, when tab visibility is set to hidden, the API returns `Hidden`, regardless of how the value was set.

ProfileUserPermission

ProfileUserPermission represents an app or system permission for a profile. Use one of these elements for each permission.

Field	Field Type	Description
`enabled`	boolean	Required. Indicates whether the permission is enabled (`true`) or disabled (`false`).
`name`	string	Required. The permission name.

Java Sample

The following sample uses picklists, profiles, record types, and a custom app:

```
public void profileSample() {
  try {
    // Create an expense report record, tab and app...
    CustomObject expenseRecord = new CustomObject();
    expenseRecord.setFullName("ExpenseReport__c");
    expenseRecord.setLabel("Expense Report");
    expenseRecord.setPluralLabel("Expense Reports");

    expenseRecord.setDeploymentStatus(DeploymentStatus.Deployed);
    expenseRecord.setSharingModel(SharingModel.ReadWrite);

    CustomField nameField = new CustomField();
    nameField.setType(FieldType.AutoNumber);
    nameField.setLabel("Expense Report Number");
    nameField.setDisplayFormat("ER-{0000}");
    expenseRecord.setNameField(nameField);

    AsyncResult[] arsExpenseRecord =
        metadataConnection.create(new Metadata[] {expenseRecord});

    Picklist expenseStatus = new Picklist();
    PicklistValue unsubmitted = new PicklistValue();
    unsubmitted.setFullName("Unsubmitted");
    PicklistValue submitted = new PicklistValue();
    submitted.setFullName("Submitted");
    PicklistValue approved = new PicklistValue();
    approved.setFullName("Approved");
    PicklistValue rejected = new PicklistValue();
    rejected.setFullName("Rejected");
    expenseStatus.setPicklistValues(new PicklistValue[] {
        unsubmitted, submitted, approved, rejected}
    );

    CustomField expenseStatusField = new CustomField();
    expenseStatusField.setFullName(
        "ExpenseReport__c.ExpenseStatus__c"
    );
    expenseStatusField.setLabel("Expense Report Status");
    expenseStatusField.setType(FieldType.Picklist);
    expenseStatusField.setPicklist(expenseStatus);
    AsyncResult[] arsStatusField =
        metadataConnection.create(new Metadata[]
            {expenseStatusField});

    CustomTab expenseTab = new CustomTab();
    expenseTab.setFullName("ExpenseReport__c");
    expenseTab.setMotif("Custom70: Handsaw");
    expenseTab.setCustomObject(true);
    AsyncResult[] arsTab =
        metadataConnection.create(new Metadata[] {expenseTab});

    CustomApplication application = new CustomApplication();
    application.setFullName("ExpenseForce");
    application.setTab(new String[] {expenseTab.getFullName()});
    AsyncResult[] arsApp =
        metadataConnection.create(new Metadata[] {application});

    // Employees and managers have the same app visibility...
    ProfileApplicationVisibility appVisibility =
        new ProfileApplicationVisibility();
    appVisibility.setApplication("ExpenseForce");
    appVisibility.setVisible(true);

    Profile employee = new Profile();
    employee.setFullName("Employee");
    employee.setApplicationVisibilities(
        new ProfileApplicationVisibility[] {appVisibility}
    );
    AsyncResult[] arsProfileEmp =
    metadataConnection.create(new Metadata[] {employee});

    Profile manager = new Profile();
    manager.setFullName("Manager");
    manager.setApplicationVisibilities(
        new ProfileApplicationVisibility[] {appVisibility}
    );
    AsyncResult[] arsProfileMgr =
        metadataConnection.create(new Metadata[] {manager});

    // But employees and managers have different access
    // to the state of the expense sheet
    RecordType edit = new RecordType();
    edit.setFullName("ExpenseReport__c.Edit");
    RecordTypePicklistValue editStatuses =
        new RecordTypePicklistValue();
    editStatuses.setPicklist("ExpenseStatus__c");
    editStatuses.setValues(new PicklistValue[]
        {unsubmitted, submitted});
    edit.setPicklistValues(new RecordTypePicklistValue[]
        {editStatuses});
    AsyncResult[] arsRecTypeEdit =
        metadataConnection.create(new Metadata[] {edit});

    RecordType approve = new RecordType();
    approve.setFullName("ExpenseReport__c.Approve");
    RecordTypePicklistValue approveStatuses =
        new RecordTypePicklistValue();
    approveStatuses.setPicklist("ExpenseStatus__c");
    approveStatuses.setValues(new PicklistValue[]
        {approved, rejected});
    approve.setPicklistValues(new RecordTypePicklistValue[]
        {approveStatuses});
    AsyncResult[] arsRecTypeApp =
        metadataConnection.create(new Metadata[] {approve});
  } catch (ConnectionException ce) {
    ce.printStackTrace();
  }
}
```

Declarative Metadata Sample Definition

The following is the definition of a profile in an organization with a custom app, custom object, record type, tab, and user permission:

```
<?xml version="1.0" encoding="UTF-8"?>
<Profile xmlns="http://soap.sforce.com/2006/04/metadata">
    <applicationVisibilities>
        <application>PubApps__Myriad_Publishing</application>
        <default>false</default>
        <visible>true</visible>
    </applicationVisibilities>
    <custom>true</custom>
    <objectPermissions>
        <object>TestWeblinks__c</object>
    </objectPermissions>
    <recordTypeVisibilities>
        <default>true</default>
        <recordType>TestWeblinks__c.My First Recordtype</recordType>
        <visible>true</visible>
    </recordTypeVisibilities>
    <tabVisibilities>
        <tab>Myriad Publications</tab>
        <visibility>DefaultOn</visibility>
    </tabVisibilities>
    <userPermissions>
        <enabled>true</enabled>
        <name>APIEnabled</name>
    </userpermissions>
</Profile>
```

Usage

When you use the `retrieve()` call to get information about profiles in your organization, the returned `.profile` files only include security settings for the other metadata types referenced in the retrieve request (except for user permissions, IP address ranges, and login hours, which are always retrieved). For example, the following `package.xml` file contains a `types` element that matches all custom objects, so the returned profiles contain object and field permissions for all custom objects in your organization, but do not include permissions for standard objects, such as Account, and standard fields.

```
<?xml version="1.0" encoding="UTF-8"?>
<Package xmlns="http://soap.sforce.com/2006/04/metadata">
    <types>
        <members>*</members>
        <name>CustomObject</name>
    </types>
   <types>
        <members>*</members>
        <name>Profile</name>
    </types>
    <version>48.0</version>
</Package>
```

The wildcard "*" on CustomObject does not match standard objects and this helps to avoid making unintended, high-impact profile changes. If you create a few custom objects in a Developer Edition organization, `retrieve()` the information, and subsequently `deploy()` the custom objects to your production organization, the profile and field-level security for all your standard objects, such as Account, and standard fields are not overwritten unless you explicitly create separate `types` elements for the standard objects or fields.

Metadata API intentionally makes it difficult to include standard fields in `retrieve()` calls to prevent unexpected profile changes. However, you can still retrieve and deploy profile permissions for custom and standard fields in standard objects, such as Account.

The next `package.xml` file allows you to return profile permissions for Account standard and custom fields. Note how the standard Account object is defined in a `types` element by specifying it as a member of a CustomObject type.

```
<?xml version="1.0" encoding="UTF-8"?>
<Package xmlns="http://soap.sforce.com/2006/04/metadata">
    <types>
        <members>Account</members>
        <name>CustomObject</name>
    </types>
   <types>
        <members>*</members>
        <name>Profile</name>
    </types>
    <version>48.0</version>
</Package>
```

The final `package.xml` file allows you to return profile permissions for the `MyCustomField__c` custom field in the Account object.

```
<?xml version="1.0" encoding="UTF-8"?>
<Package xmlns="http://soap.sforce.com/2006/04/metadata">
   <types>
        <members>Account.MyCustomField__c</members>
        <name>CustomField</name>
   </types>
```

```
    <types>
        <members>*</members>
        <name>Profile</name>
    </types>
    <version>48.0</version>
</Package>
```

Wildcard Support in the Manifest File

This metadata type supports the wildcard character * (asterisk) in the package.xml manifest file. For information about using the manifest file, see Deploying and Retrieving Metadata with the Zip File.

ProfileActionOverride

Represents an override of an ActionOverride by a user profile. You can use it to override an ActionOverride on a standard Home tab or object record page in Lightning Experience. When a user logs in with a profile, a matching ProfileActionOverride assignment takes precedence over existing overrides for the Home tab or record page specified in ActionOverride. In API versions 39.0 to 44.0, you can access ProfileActionOverride by accessing its encompassing CustomApplication or Profile metadata types. In API version 45.0 and later, you can access ProfileActionOverride only by accessing its encompassing CustomApplication.

Note: ProfileActionOverrides aren't supported in packaging. They are supported in change sets, but you have to add them manually.

File Suffix and Directory Location

Profile-based action overrides are defined as part of a custom application or profile.

Version

ProfileActionOverrides are available in API version 39.0 and later.

ProfileActionOverride can be defined on Profile or CustomApplication for API version 39.0 to 44.0. In API version 45.0 and later, ProfileActionOverride must be defined for CustomApplication instead. Beginning with API version 45.0, Home page assignments related to user profile must also have a corresponding app assignment because more granular Home page assignments are supported. As a result, ProfileActionOverride is defined for CustomApplication rather than Profile.

Fields

Field Name	Field Type	Description
actionName	string	The name of the action. The only valid values are Tab and View. If pageOrSobjectType is standard-home, this field must be Tab. The Tab action is supported only when ProfileActionOverride is being specified as part of a Profile in API version 39.0 to 44.0. In API version 45.0 and later, this action is supported only when ProfileActionOverride is being specified as part of a CustomApplication, pageOrSobjectType is standard-home, and this field is Tab.

Field Name	Field Type	Description
		If `pageOrSobjectType` is `record-home`, this field must be `View`. The `View` action is supported only when ProfileActionOverride is being specified as part of a CustomApplication.
`content`	string	Read-only. Represents the name of the Lightning page being used as the override.
`formFactor`	FormFactor (enumeration of type string)	The size of the page being overridden. The `Large` value represents the Lightning Experience desktop environment.
`pageOrSobjectType`	string	The name of the page being overridden. The only valid values are `record-home` and `standard-home`. If the `actionName` is `Tab`, this field must be `standard-home`
`recordType`	string	The record type associated with the override. If `pageOrSobjectType` is `standard-home`, this field must be `null`. This field is required when `actionName` is set to `View`.
`type`	ActionOverrideType (enumeration of type string)	Read-only. The type of action override. The only valid value is `flexipage`.

Declarative Metadata Sample Definition

You can define a ProfileActionOverride like this.

```
<CustomApplication xmlns="http://soap.sforce.com/2006/04/metadata">
    <profileActionOverrides>
        <actionName>View</actionName>
        <content>CustomObjectFlexiPage</content>
        <formFactor>Large</formFactor>
        <pageOrSobjectType>TestObj__c</pageOrSobjectType>
        <type>Flexipage</type>
        <profile>standard</profile>
        <recordType>TestObj__c.TestRecordType</recordType>
    </profileActionOverrides>
    <defaultLandingTab>standard-home</defaultLandingTab>
    <formFactors>Large</formFactors>
    <label>My Custom App</label>
    <tab>standard-Account</tab>
    <tab>standard-Opportunity</tab>
    <uiType>Lightning</uiType>
    <navType>Standard</navType>
</CustomApplication>
```

Here is an example `package.xml`.

```
<?xml version="1.0" encoding="UTF-8"?>
<Package xmlns="http://soap.sforce.com/2006/04/metadata">
    <types>
```

```
        <members>MyCustomApp</members>
        <name>CustomApplication</name>
    </types>
    <version>39.0</version>
</Package>
```

Wildcard Support in the Manifest File

This metadata type doesn't support the wildcard character `*` (asterisk) in the `package.xml` manifest file. For information about using the manifest file, see Deploying and Retrieving Metadata with the Zip File.

ProfilePasswordPolicy

Represents a profile's password policies. Profile password policies override org-wide password policies for that profile's users. Use ProfilePasswordPolicy to retrieve password policies for a given profile. This type extends the Metadata metadata type and inherits its `fullName` field.

File Suffix and Directory Location

ProfilePasswordPolicy components have the suffix `.profilePasswordPolicy` and are stored in the `profilePasswordPolicies` folder.

Version

ProfilePasswordPolicy components are available in API version 40.0 and later.

Fields

Field Name	Field Type	Description
`forgotPasswordRedirect`	boolean	If `true`, reset password links in forgot password emails don't immediately expire the first time they're clicked. Instead, the links stay active until a user confirms the password reset request on an interstitial page. The default value is `false`. This field is available in API version 43.0 and later.
`lockoutInterval`	int	Required. The duration of the login lockout, in minutes. If users are locked out, they must wait until the lockout period expires. Valid values: `0`, `15`, `30`, `60`.
`maxLoginAttempts`	int	Required. The number of times a user can enter a wrong password before getting locked out. Valid values: `0`, `3`, `5`, `10`.
`minimumPasswordLength`	int	Required. Minimum number of characters required for a password. Valid values: `5–50`.
`minimumPasswordLifetime`	boolean	If `true`, a user cannot change a password more than once in a 24-hour period.

Field Name	Field Type	Description
`obscure`	boolean	If `true`, answers to security questions are hidden as the user types.
`passwordComplexity`	int	Required. Level of complexity required for the character types in a user's password. • If `0`, the password can contain any type of character. • If `1`, the password must contain at least one alphabetic character and 1 number. • If `2`, the password must contain at least one alphabetic character, one number, and one of the following special characters: ! # $ % - _ = + < >. • If `3`, the password must contain at least one number, one uppercase letter, and one lowercase letter. • If `4`, the password must contain at least one number, one uppercase letter, one lowercase letter, and one of the following special characters: ! # $ % - _ = + < >.
`passwordExpiration`	int	Required. Number of days until user passwords expire and must be changed. If set to `0`, the password never expires. Valid values: `0`, `30`, `60`, `90`, `365`.
`passwordHistory`	int	Required. Number of previous passwords to save. Saving passwords is required to ensure that users reset their password to a new, unique password. This value must be set before a password reset succeeds. If `0`, `passwordExpiration` must be set to `0`.
`passwordQuestion`	int	Required. If set to `1`, the answer to the password hint cannot contain the password itself. If `0`, the answer has no restrictions.
`profile`	string	Required. Name of the user profile.

Declarative Metadata Sample Definition

The following is an example of a ProfilePasswordPolicy component.

```
<?xml version="1.0" encoding="UTF-8"?>
<ProfilePasswordPolicy xmlns="http://soap.sforce.com/2006/04/metadata">
    <forgotPasswordRedirect>true</forgotPasswordRedirect>
    <lockoutInterval>30</lockoutInterval>
    <maxLoginAttempts>0</maxLoginAttempts>
    <minimumPasswordLength>7</minimumPasswordLength>
    <minimumPasswordLifetime>false</minimumPasswordLifetime>
    <obscure>false</obscure>
    <passwordComplexity>1</passwordComplexity>
    <passwordExpiration>0</passwordExpiration>
    <passwordHistory>0</passwordHistory>
    <passwordQuestion>1</passwordQuestion>
    <profile>platformportal</profile>
</ProfilePasswordPolicy>
```

Wildcard Support in the Manifest File

This metadata type supports the wildcard character `*` (asterisk) in the `package.xml` manifest file. For information about using the manifest file, see Deploying and Retrieving Metadata with the Zip File.

ProfileSessionSetting

Represents a profile's session settings. Use ProfileSessionSetting to retrieve the session settings for a given profile. This type extends the Metadata metadata type and inherits its `fullName` field.

File Suffix and Directory Location

ProfileSessionSetting components have the suffix `.profileSessionSetting` and are stored in the `profileSessionSettings` folder.

Version

ProfileSessionSetting components are available in API version 40.0 and later.

Fields

Field Name	Field Type	Description
`forceLogout`	boolean	Required. If `true`, when the session times out, users are logged out immediately and redirected to a default or custom logout URL.
`profile`	string	Required. Name of the user profile.
`requiredSessionLevel`	SessionSecurityLevel	Session security level.
`sessionPersistence`	boolean	Beta. If `true`, keepusers logged in to their community until the session times out—even if they close their browser. Use `sessionPersistence` to reduce how often users must log in to their community. Applies only to the External Identity profile.
`sessionTimeout`	int	Required. Specifies how many minutes of inactivity elapse before a user's authenticated session times out. At the end of the session, the user must log in again. This session timeout value applies to users of the profile and overrides the org-wide timeout value. Changes to the org-wide timeout value don't apply to users of this profile. Valid values: `15`, `30`, `60`, `120`, `240`, `480`, `720`, `1440`.
`sessionTimeoutWarning`	boolean	Required. If `true`, the user receives a warning when the session is about to expire.

SessionSecurityLevel

Session security levels control access to certain types of resources based on the type of authentication used for logging in to the current session. For example, username and password authentication requires the `standard` session security level. Two-factor authentication requires `HIGH_ASSURANCE`.

Field Name	Field Type	Description
`SessionSecurityLevel`	(enumeration of type string)	User's security level for the current session. • The `HIGH_ASSURANCE` security level for this session meets the High Assurance requirements set in the org's session settings under Session Security Levels. • The `STANDARD` security level for this session meets the Standard requirements set in the org's session settings under Session Security Levels. • The `LOW` level is not available or used in the Salesforce UI. It is used at the API level, but users assigned to this level experience unpredictable and reduced functionality.

Declarative Metadata Sample Definition

The following is an example of a ProfileSessionSetting component.

```xml
<?xml version="1.0" encoding="UTF-8"?>
<ProfileSessionSetting xmlns="http://soap.sforce.com/2006/04/metadata">
    <forceLogout>false</forceLogout>
    <profile>platformportal</profile>
    <requiredSessionLevel>HIGH_ASSURANCE</requiredSessionLevel>
    <sessionTimeout>1440</sessionTimeout>
    <sessionTimeoutWarning>false</sessionTimeoutWarning>
</ProfileSessionSetting>
```

Wildcard Support in the Manifest File

This metadata type supports the wildcard character `*` (asterisk) in the `package.xml` manifest file. For information about using the manifest file, see Deploying and Retrieving Metadata with the Zip File.

Prompt

Represents the metadata associated with in-app guidance prompts. Use prompts to display announcements, training, or news to users within the app. Choose to add action button that links to a URL of your choice. Track view and button click completes.This type extends the Metadata metadata type and inherits its `fullName` field.

File Suffix and Directory Location

Prompt components have the suffix `prompt` and are stored in the `prompts` folder.

Version

Prompt components are available in API version 46.0 and later.

Special Access Rules

Admins have access to add or edit prompts. For non-admin users, assign the Manage Prompts and Modify Metadata user permission. Everyone can see the In-App Guidance setup page.

Packaging Prompts

See Creating Managed Packages in Salesforce Help for more information.

See Considerations for Prompts in Lightning Experience in Salesforce Help for more information about installing and managing prompt packages and about editing and cloning prompts installed from packages.

Unmanaged packages for prompts isn't supported.

When orgs install prompts from packages, the prompts will be the same publish state as indicated by the `IsPublished` field. For example, if the package prompt is active, it will also be active when installed by the user.

If the package includes a custom profile or permission that isn't part of the installer's Salesforce org, the prompt is installed, but it won't include those items. For example, you install a prompt with several custom profiles not included in your org. The prompts are installed without those custom profiles.

If the package includes a standard app that isn't part of the installer's Salesforce org, the prompts are installed, but aren't usable.

Fields

Field Name	Field Type	Description
`masterLabel`	string	Required. The master label for the prompt. Maximum of 80 characters.
`promptVersions`	PromptVersion[]	A list of prompt entries. Each entry represent a different prompt.

PromptVersion

A list of prompt entries. Each entry represents a different prompt.

Field Name	Field Type	Description
`actionButtonLabel`	string	Label for the action button. Maximum of 25 characters.
`actionButtonLink`	string	URL for the action button. Maximum of 1,000 characters. You can't use the `GROUP BY` option in a SOQL query for this field.
`body`	string	Required. Body content of the prompt. For floating prompts, there's a maximum of 120 characters. For docked prompts, there's a maximum of 4000 characters.
`customApplication`	string	Internal use only. No data is populated for this field.
`delayDays`	int	Required if recurrences are scheduled. Number of days in between occurrences of the prompt.

Field Name	Field Type	Description
description	string	Description of the prompt. Maximum of 255 characters.
dismissButtonLabel	string	Label for the dismiss button of a floating prompt. Maximum of 15 characters.
displayPosition	PromptDisplayPosition (enumeration of type string)	Indicates the position of the floating prompt on the page. Valid values are: • BottomCenter • BottomLeft • BottomRight • TopCenter • TopLeft • TopRight
displayType	PromptDisplayType (enumeration of type string)	Required. Indicates the type of prompt. Valid values are: • DockedComposer, which is the docked prompt • FloatingPanel, which is the floating prompt
endDate	date	Indicates the date to stop showing the prompt.
header	string	Label for the header of the docked prompt. This is the label contained in the window's browser bar. Maximum of 36 characters.
indexWithIsPublished	string	Used by Salesforce for efficient querying.
indexWithoutIsPublished	string	Used by Salesforce for efficient querying.
isPublished	boolean	Indicates if a prompt is active true or not false.
masterLabel	string	Required. The master label for the prompt.
publishedByUser	string	Internal use only. No data is populated for this field.
publishedDate	date	Indicates the date the prompt was activated. If the prompt is part of a package, this is the date when the package was installed.
shouldDisplayActionButton	boolean	Required. Indicates if an action button is included in the prompt true or not false.
shouldIgnoreGlobalDelay	boolean	Indicates if the prompt ignores the global time delay between prompts and instead shows on page load true or not false. This field is available in API version 48.0 and later.
startDate	date	Required. Indicates the date to start showing the prompt.
targetAppDeveloperName	string	Required. The app's developer name where the prompt appears.
targetAppNamespacePrefix	string	The app's namespace prefix where the prompt appears.
targetPageKey1	string	Required. Used by Salesforce to identity the prompt page location along with targetPageKey2 and targetPageType.
targetPageKey2	string	Used by Salesforce to identity the prompt page location along with targetPageKey1 and targetPageType.

Field Name	Field Type	Description
`targetPageType`	string	Required. Used by Salesforce to identity the prompt page location along with `targetPageKey1` and `targetPageKey2`.
`timesToDisplay`	int	Required if recurrences are scheduled. Maximum number of times to display the prompt (that is, the number of occurrences). Salesforce detects if the user interacts with (or ignores) the prompt to determine if we should show the prompt again or cancel scheduled recurrences. This might run counter to the number of occurrences scheduled. Maximum value of 30.
`title`	string	Required. The label for the title of the prompt. Maximum of 36 characters.
`uiFormulaRule`	UiFormulaRule	A set of one or more permission filters that define the conditions under which the prompt displays on the page. If the rule evaluates to `true`, the prompt displays on the page. If `false`, it doesn't display. If this field is `null`, the prompt displays by default.
`userAccess`	PromptUserAccess (enumeration of type string)	Required. Indicates which permissions can see the prompt. Valid values are: • `Everyone`, which indicates that there's no permission restrictions on who can see the prompt • `SpecificPermissions`, which indicates that only users with all the specific user permissions specified can see the prompt
`userProfileAccess`	PromptUserProfileAccess (enumeration of type string)	Indicates which profiles can see the prompt. This field is available in API version 48.0 and later. Valid values are: • `Everyone`, which indicates that there are no profile restrictions on who can see the prompt • `SpecificProfiles`, which indicates that users with any of the specified user profiles can see the prompt
`versionNumber`	int	Required. The number remains `1` since multiple versions aren't saved in the org.
`videoLink`	string	The URL for the video in a docked prompt. Maximum of 1,000 characters. This field is available in API version 48.0 and later. To find the embed code for a video, follow the instructions from the video host website. Usually the steps can be found by searching for the name of the website and "embed video." For example, here's what the embed code looks like for YouTube: `<iframe width="560" height="315" src="https://www.youtube.com/embed/Ko-gcObzTVo" frameborder="0" allow="accelerometer; autoplay; encrypted-media; gyroscope; picture-in-picture" allowfullscreen></iframe>` Then, you would enter the URL found in the `src` attribute. For the example used, enter *`https://www.youtube.com/embed/Ko-gcObzTVo`*.

UiFormulaRule

A set of one or more filters that define the conditions under which a prompt displays on a Lightning page.

Field Name	Field Type	Description
`booleanFilter`	string	Specifies the AND filter condition.
`criteria`	UiFormulaCriterion[]	List of one or more filters that, when evaluated, determine prompt visibility.

UiFormulaCriterion

A single filter that, when evaluated, helps define prompt visibility on a Lightning page.

Field Name	Field Type	Description
`leftValue`	string	Required. The field upon which the filter is based. Only standard and custom permissions can be included. You can use these expressions in the `leftValue` field when setting filters for prompt visibility. • `{!$Permission.CustomPermission.`***`permissionName`***`}`—Use this expression to control prompt visibility based on the custom permissions of the user viewing the Lightning page. Supported for app, Home, and record pages only. • `{!$Permission.StandardPermission.`***`permissionName`***`}`—Use this expression to control prompt visibility based on the standard permissions of the user viewing the Lightning page. Supported for app, Home, and record pages only. • `{!ENCODED:{!ID:$`***`User.Profile.Key`***`}}`—Use this expression to control prompt visibility based on the custom or standard profile of the user viewing the Lightning page. Available in API Version 48.0 and later.
`operator`	string	Required. Defines the operator used to filter the data. Valid value is `EQUAL`.
`rightValue`	string	Specifies if you want to evaluate the prompt's visibility for permissions or the name of the profile. • For permissions, use `true`. • For profiles, use the name of the profile. Available in API Version 48.0 and later. For example, `Standard`or `custom_regionalsales`.

Declarative Metadata Sample Definition

The following is an example of a Prompt component.

```
<?xml version="1.0" encoding="UTF-8"?>
<Prompt xmlns="http://soap.sforce.com/2006/04/metadata">
    <masterLabel>Prompt Master Label</masterLabel>
    <promptVersions>
        <actionButtonLabel>Learn How</actionButtonLabel>

<actionButtonLink>https://trailhead.salesforce.com/en/content/learn/modules/scrum-and-kanban-at-salesforce/learn-about-kanban</actionButtonLink>

        <body>Explore how the Path and the Kanban view can help you track, manage, and
update your records.</body>
        <delayDays>1</delayDays>
        <description>Kanban floating prompt</description>
        <dismissButtonLabel>OK</dismissButtonLabel>
        <displayPosition>TopLeft</displayPosition>
        <displayType>FloatingPanel</displayType>
        <endDate>2019-03-11</endDate>
        <isPublished>true</isPublished>
        <masterLabel>Prompt Master Label</masterLabel>
        <publishedDate>2019-03-11</publishedDate>
        <shouldDisplayActionButton>false</shouldDisplayActionButton>
        <shouldIgnoreGlobalDelay>false</shouldIgnoreGlobalDelay>
        <startDate>2019-03-11</startDate>
        <targetAppDeveloperName>LightningSales</targetAppDeveloperName>
        <targetAppNamespacePrefix>standard</targetAppNamespacePrefix>
        <timesToDisplay>3</timesToDisplay>
        <title>Get on the Path to Success</title>
        <userAccess>SpecificPermissions</userAccess>
        <userProfileAccess>SpecificProfiles</userProfileAccess>
        <versionNumber>1</versionNumber>
        <videolink>https://www.youtube.com/embed/Ko-gcObzTVo</videolink>
        <uiFormulaRule>
            <booleanFilter>(1 AND 2 AND 3) AND (4 OR 5)</booleanFilter>
            <criteria>
                <leftValue>{!$Permission.StandardPermission.ActivitiesAccess}</leftValue>

                <operator>EQUAL</operator>
                <rightValue>TRUE</rightValue>
            </criteria>
            <criteria>
               <leftValue>{!$Permission.StandardPermission.ContentWorkspaces}</leftValue>

                <operator>EQUAL</operator>
                <rightValue>TRUE</rightValue>
            </criteria>
            <criteria>
                <leftValue>{!$Permission.CustomPermission.MyCustomPerm}</leftValue>
                <operator>EQUAL</operator>
                <rightValue>TRUE</rightValue>
            </criteria>
            <criteria>
                <leftValue>{!ENCODED:{!ID:$User.Profile.Key}}</leftValue>
                <operator>EQUAL</operator>
                <rightValue>Standard</rightValue>
            </criteria>
            <criteria>
                <leftValue>{!ENCODED:{!ID:$User.Profile.Key}}</leftValue>
                <operator>EQUAL</operator>
                <rightValue>custom_mysysadmin</rightValue>
            </criteria>
        </uiFormulaRule>
    </promptVersions>
</Prompt>
```

The following is an example `package.xml` that references the previous definition.

```
<?xml version="1.0" encoding="UTF-8"?>
<Package xmlns="http://soap.sforce.com/2006/04/metadata">
    <types>
        <members>*</members>
        <name>Prompt</name>
    </types>
    <version>46.0</version>
</Package>
```

Wildcard Support in the Manifest File

This metadata type supports the wildcard character `*` (asterisk) in the `package.xml` manifest file. For information about using the manifest file, see Deploying and Retrieving Metadata with the Zip File.

Queue

Represents a holding area for items before they are processed.

Declarative Metadata File Suffix and Directory Location

The file suffix for queue components is `.queue` and components are stored in the `queues` directory of the corresponding package directory. This component supports cases, leads, service contracts (if Entitlements are enabled), and custom objects.

Version

Queue components are available in API version 24.0 and later.

Fields

This metadata type represents the valid values that define a queue:

Field Name	Field Type	Description
`doesSendEmailToMembers`	boolean	Indicates whether emails are sent to queue members (`true`) or not (`false`) when a new record is added to the queue.

Field Name	Field Type	Description
`email`	string	The email address of the queue owner.
`name`	string	Required. The name of the queue. Corresponds to **Label** in the user interface.
`queueMembers`	QueueMembers[]	Represents queue members added to the queue. Members can be added directly or selected by roles and public groups they belong to. Available in API version 42.0 and later.
`queueRoutingConfig`	string	Routing configuration name. Applies to orgs that use Omni-Channel with a routing configuration. Available in API version 42.0 and later.
`queueSobject`	QueueSobject[]	Indicates the supported entity types.

QueueMembers

Represents queue members added to the queue. Members can be added directly as users or selected by the roles and public groups they belong to. Available in API version 42.0 and later.

Field Name	Field Type	Description
`publicGroups`	PublicGroups[]	Represents public groups in the org. Public groups are optionally used to add queue members.
`roleAndSubordinates`	RoleAndSubordinates[]	Represents roles and their subordinates in the org's role hierarchy, including customer and partner roles. Roles and their subordinate hierarchy are optionally used to add queue members.
`roleAndSubordinatesInternal`	RoleAndSubordinatesInternal[]	Represents internal roles and their subordinates in the org's role hierarchy, excluding customer and partner roles. Roles and their subordinate hierarchy are optionally used to add queue members.
`roles`	Roles[]	Represents roles in the org. Roles are optionally used to add queue members.
`users`	Users[]	Represents users in the org. Users can be added directly as queue members.

PublicGroups

Represents public groups in the org. Public groups are optionally used to add queue members. Available in API version 42.0 and later.

Field Name	Field Type	Description
`publicGroup`	string	Represents a public group.

RoleAndSubordinates

Represents roles and their subordinates in the org's role hierarchy, including customer and partner roles. Roles and their subordinate hierarchy can be used to add queue members. Available in API version 42.0 and later.

Field Name	Field Type	Description
`roleAndSubordinate`	string	Represents a role and its subordinates, including customer and partner roles.

RoleAndSubordinatesInternal

Represents internal roles and their subordinates in the org's role hierarchy, excluding customer and partner roles. Roles and their subordinate hierarchy can be used to add queue members. Available in API version 42.0 and later.

Field Name	Field Type	Description
`roleAndSubordinateInternal`	string	Represents a role and its subordinates, excluding customer and partner roles.

Roles

Represents roles in the org. Roles can be used to add queue members. Available in API version 42.0 and later.

Field Name	Field Type	Description
`role`	string	Represents a role.

Users

Represents users in the org. Users can be added directly as queue members. Available in API version 42.0 and later.

Field Name	Field Type	Description
`user`	string	Represents a user.

QueueSobject

QueueSobject represents an entity type that the queue supports.

Field Name	Field Type	Description
`sobjectType`	string	Valid values are: • `Case` • `ContactRequest` • `Lead`

Field Name	Field Type	Description
		- ServiceContract - Task (Available in API version 48.0 and later.) - Custom objects (such as ObjA_c)

Declarative Metadata Sample Definition

The following is the definition of a queue, which supports Case, Lead, and a custom object named ObjA.

```
<?xml version="1.0" encoding="UTF-8"?>
<Queue xmlns="http://soap.sforce.com/2006/04/metadata">
    <doesSendEmailToMembers>true</doesSendEmailToMembers>
    <email>member@company.com</email>
    <fullName>Your Name</fullName>
    <name>memberQueue</name>
    <queueSobject>
        <sobjectType>Case</sobjectType>
    </queueSobject>
    <queueSobject>
        <sobjectType>Lead</sobjectType>
    </queueSobject>
    <queueSobject>
        <sobjectType>ObjA__c</sobjectType>
    </queueSobject>
</Queue>
```

Here's another definition of a queue containing queue members added directly or via public groups and roles. Queries retrieve values using the DeveloperName field, not the Name field, so that the returned names are unique. The query also appends letters to the end of duplicate names, so these groups and roles can be referred to independently.

```
<?xml version="1.0" encoding="UTF-8"?>
<Queue xmlns="http://soap.sforce.com/2006/04/metadata">
    <doesSendEmailToMembers>false</doesSendEmailToMembers>
    <name>queue1</name>
    <members>
        <publicGroups>
            <publicGroup>All Internal Users</publicGroup>
        <publicGroups>
        <queueRoleAndSubordinates>
            <queueRoleAndSubordinate>role1</queueRoleAndSubordinate>
            <queueRoleAndSubordinate>role2</queueRoleAndSubordinate>
            <queueRoleAndSubordinate>role3</queueRoleAndSubordinate>
        </queueRoleAndSubordinates>
        <roles>
            <role>role1</role>
        </roles>
        <users>
            <user>s@sm.com</user>
            <user>std@sm.com</user>
        </users>
    </members>
    <queueRoutingConfig>my_omni_routing_config</queueRoutingConfig>
    <queueSobject>
        <sobjectType>Case</sobjectType>
    </queueSobject>
    <queueSobject>
        <sobjectType>Lead</sobjectType>
    </queueSobject>
</Queue>
```

Wildcard Support in the Manifest File

This metadata type supports the wildcard character `*` (asterisk) in the `package.xml` manifest file. For information about using the manifest file, see Deploying and Retrieving Metadata with the Zip File.

QueueRoutingConfig

Represents the settings that determine how work items are routed to agents.This type extends the Metadata metadata type and inherits its `fullName` field.

File Suffix and Directory Location

ServicePresenceStatus components have the suffix `.queueRoutingConfig` and are stored in the `queueRoutingConfigs` folder.

Version

QueueRoutingConfig components are available in API version 44.0 and later.

Special Access Rules

This type is available only if Omni-Channel is enabled in your org.

Fields

Field Name	Field Type	Description
`capacityPercentage`	double	The percentage of an agent’s capacity for work items that’s consumed by a specific type of work item from this service channel. For example, you might give phone calls a capacity percentage of 100. If an agent receives a phone call, the agent won’t receive new work items until the call ends, because at that point the agent’s capacity will have reached 100%.
`capacityWeight`	double	The amount of an agent’s capacity for work items that’s consumed by a work item from this service channel. For example, if an agent has a capacity of 6, and cases are assigned a capacity weight of 2, an agent

Field Name	Field Type	Description
		can be assigned up to 3 cases before the agent is at capacity and can't receive new work items.
dropAdditionalSkillsTimeout	int	The number of seconds to elapse before additional skills are dropped from Omni-Channel routing. In skills-based routing, you can set some skills to **Additional Skill**. After the timeout elapses, a skill marked as **Additional Skill** is dropped from Omni-Channel routing and the case is routed to the best-matched agent, even if the agent doesn't have all the skills.
isAttributeBased	boolean	Indicates whether this routing configuration is used with attribute setup for skills-based routing (`true`) or not (`false`).
label	string	Required. The label of the presence status.
pushTimeout	int	The number of seconds set for push timeout. `0` is returned when push timeout isn't enabled.
queueOverflowAssignee	string	The ID of the queue that's set as the Overflow Assignee.
routingModel	RoutingModel (enumeration of type string)	Required. The routing type that determines how work items are routed (pushed) to agents. Possible values are: • `LEAST_ACTIVE` • `MOST_AVAILABLE` • `EXTERNAL_ROUTING`
routingPriority	int	Required. The priority in which work items from the service channels that are related to this routing configuration are routed to agents. Work items from routing configurations that have lower priority values (for example, 0) are routed to agents first.
userOverflowAssignee	string	The ID of the user that's set as the Overflow Assignee.

Declarative Metadata Sample Definition

The following is an example of a QueueRoutingConfig component.

```
<?xml version="1.0" encoding="UTF-8"?>
<QueueRoutingConfig xmlns="http://soap.sforce.com/2006/04/metadata">
    <capacityWeight>1.0</capacityWeight>
    <label>Case Routing</label>
    <pushTimeout>120</pushTimeout>
    <queueOverflowAssignee>queueOverflow</queueOverflowAssignee>
    <routingModel>LEAST_ACTIVE</routingModel>
    <routingPriority>1</routingPriority>
</QueueRoutingConfig>
```

The following is an example `package.xml` that references the previous definition.

```
<?xml version="1.0" encoding="UTF-8"?>
<Package xmlns="http://soap.sforce.com/2006/04/metadata">
```

```
    <types>
        <members>*</members>
        <name>QueueRoutingConfig</name>
    </types>
    <version>44.0</version>
</Package>
```

Wildcard Support in the Manifest File

This metadata type supports the wildcard character `*` (asterisk) in the `package.xml` manifest file. For information about using the manifest file, see Deploying and Retrieving Metadata with the Zip File.

QuickAction

Represents a specified create or update quick action for an object that then becomes available in the Chatter publisher. For example, you can create an action that, on the detail page of an account, allows a user to create a contact related to that account from the Chatter feed on that page. QuickAction can be created on objects that allow custom fields.

The parent objects supported include:

- Account
- Campaign
- Case
- Contact
- ContentNote
- Custom objects
- Group
- Lead
- Opportunity

File Suffix and Directory Location

QuickAction components have the suffix `quickAction` and are stored in the `quickActions` folder.

Version

QuickAction components are available in API version 28.0 and later.

Fields

Field Name	Field Type	Description
`canvas`	string	If the custom action invokes a Canvas app, the app name. Returns the fully qualified name of the Canvas app in the format `<namespace>__<dev_name>`, if the quick action type is `Canvas`; otherwise, returns `null`.

Field Name	Field Type	Description
		This field is available in API version 29.0 and later.
description	string	The description of the action.
fieldOverrides	FieldOverride on page 766[]	The specific field that may be overridden within a QuickAction.
flowDefinition	string	If the custom action invokes a flow, this field represents the API name of the flow. Otherwise, this field is `null`.
height	int	If a custom action is created, this field represents the height in pixels of the action pane.
icon	string	The icon used to identify the action. API version 32.0 and later returns different icons than in earlier API versions.
isProtected	boolean	Indicates whether this component is protected (`true`) or not (`false`). Protected components cannot be linked to or referenced by components created in the installing organization.
label	string	Identifies the action and displays to users. This is also the default identifier used for the API and managed packages.
lightningComponent	string	If the custom action invokes a Lightning component, this field represents the fully qualified name of the component. Otherwise, this field is `null`. Available in API version 38.0 and later.
optionsCreateFeedItem	boolean	Required. Indicates whether successful completion of the action creates a feed item (`true`) or not (`false`). Applies only to Create Record, Update Record, and Log a Call quick action types. Available in API version 36.0 and later.
page	string	If the custom action invokes a Visualforce page, this field identifies the page.
quickActionLayout	QuickActionLayout	The layout of fields on the action.
standardLabel	QuickActionLabel (enumeration of type string)	Specifies the standard label to use for the action. The valid values are: • `AddRecord` • `AddMember` • `ChangeDueDate` • `ChangePriority` • `ChangeStatus` • `CreateNew` • `CreateNewRecordType` (For example, a label with something like "Create New Idea") • `Defer`

Field Name	Field Type	Description
		- EditDescription - Escalate - EscalateToRecord - Forward (Available in API version 42.0 and later) - LogACall - LogANote - ModifyAppointment (Available in API version 47.0 and later) - New (A new record) - NewChild (A new child record) - NewChildRecordType - NewRecordType (For example, a label with something like "New Idea") - OfferFeedback - Quick (A quick record) - QuickRecordType - Reply (Available in API version 42.0 and later) - ReplyAll (Available in API version 42.0 and later) - RequestFeedback - SendEmail (This value is available in API version 31.0 and later.) - Update
successMessage	string	The message that displays to the user upon successful completion of the action. Available in API version 36.0 and later.
targetObject	string	The object for which the action is created and performed. For example, you can create an action that, on the detail page of an account, allows a user to create a contact related to that account from the Chatter feed on that page. In this case, Contact is the targetObject.
targetParentField	string	The parent object type of the action. Links the target object to the parent object. For example, use Account if the target object is Contact and the parent object is Account.
targetRecordType	string	Specifies which record type to create. Valid values are: - Business Account - Person Account - Master

Field Name	Field Type	Description
type	QuickActionType (enumeration of type string)	Required. The type of quick action. Valid values are: • Canvas • Create • Flow (This value is available as a Beta in API version 41.0 and later) • LightningComponent (This value is available in API version 38.0 and later.) • LogACall • Post • SendEmail (This value is available in API version 31.0 and later.) • SocialPost • Update • VisualforcePage
width	int	If a custom action is created, this field represents the width in pixels of the action pane.

FieldOverride

Represents the field names and their respective formulas and literal values that comprise predefined value settings for a QuickAction. If a field on an action has both a predefined value and a default value set, the action uses the predefined value, not the default value. A formula value takes precedence over a literal value if both are defined.

Field Name	Field Type	Description
field	string	Required. The name of the field to allow predefined values on.
formula	string	Specifies the formula to use when setting a field's predefined value. Supported for single-select picklists as of API version 43.0.
literalValue	string	Supported for picklists only. Specifies the literal value of the field defined from values in the picklist. Corresponds to the Specific Value field in the predefined value UI.

QuickActionLayout

The layout of fields on the action. There is no hard limit to the number of fields you can add to an action layout. However, for optimum usability, we recommend a maximum of eight fields. Adding more than 20 fields can severely affect user efficiency.

Field Name	Field Type	Description
layoutSectionStyle	LayoutSectionStyle (enumeration of type string)	Required. The type of layout structure used. The valid values are: • TwoColumnsTopToBottom • TwoColumnsLeftToRight • OneColumn

Field Name	Field Type	Description
		- CustomLinks
quickActionLayoutColumns	QuickActionLayoutColumn[]	Specifies columns in a QuickActionLayout.

QuickActionLayoutColumn

A column defined for a QuickActionLayout.

Field Name	Field Type	Description
quickActionLayoutItems	QuickActionLayoutItem []	Specifies row items in a QuickActionLayoutColumn.

QuickActionLayoutItem

A row item comprised of fields and defined for a QuickActionLayoutColumn.

Field Name	Field Type	Description
emptySpace	boolean	Controls if this layout item is a blank space (true) or not (false).
field	string	Represents a specific field in QuickActionLayoutItem. There is no hard limit to the number of fields you can add to an action layout. However, for optimum usability, we recommend a maximum of eight fields. Adding more than 20 fields can severely affect user efficiency.
uiBehavior	UiBehavior (enumeration of type string)	Specifies user input behavior for specific fields in QuickActionLayoutItem. The valid values are: - Edit - Required - Readonly

Declarative Metadata Sample Definition

The following is an example of a QuickAction component:

```
<?xml version="1.0" encoding="UTF-8"?>
<QuickAction xmlns="http://soap.sforce.com/2006/04/metadata">
    <description>testActionDefinitionTypesCreateTask</description>
    <label>testActionDefinitionTypesCreateTask</label>
    <optionsCreateFeedItem>true</optionsCreateFeedItem>
    <quickActionLayout>
        <layoutSectionStyle>TwoColumnsLeftToRight</layoutSectionStyle>
        <quickActionLayoutColumns>
            <quickActionLayoutItems>
```

```
                <emptySpace>false</emptySpace>
                <field>OwnerId</field>
                <uiBehavior>Required</uiBehavior>
            </quickActionLayoutItems>
            <quickActionLayoutItems>
                <emptySpace>false</emptySpace>
                <field>WhoId</field>
                <uiBehavior>Edit</uiBehavior>
            </quickActionLayoutItems>
            <quickActionLayoutItems>
                <emptySpace>false</emptySpace>
                <field>WhatId</field>
                <uiBehavior>Edit</uiBehavior>
            </quickActionLayoutItems>
            <quickActionLayoutItems>
                <emptySpace>false</emptySpace>
                <field>ActivityDate</field>
                <uiBehavior>Edit</uiBehavior>
            </quickActionLayoutItems>
            <quickActionLayoutItems>
                <emptySpace>false</emptySpace>
                <field>Subject</field>
                <uiBehavior>Edit</uiBehavior>
            </quickActionLayoutItems>
            <quickActionLayoutItems>
                <emptySpace>false</emptySpace>
                <field>Status</field>
                <uiBehavior>Required</uiBehavior>
            </quickActionLayoutItems>
            <quickActionLayoutItems>
                <emptySpace>false</emptySpace>
                <field>Priority</field>
                <uiBehavior>Required</uiBehavior>
            </quickActionLayoutItems>
        </quickActionLayoutColumns>
        <quickActionLayoutColumns/>
    </quickActionLayout>
    <successMessage>This is a success message</successMessage>
    <targetObject>Task</targetObject>
    <targetParentField>What</targetParentField>
    <type>Create</type>
</QuickAction>
```

Wildcard Support in the Manifest File

This metadata type supports the wildcard character `*` (asterisk) in the `package.xml` manifest file. For information about using the manifest file, see Deploying and Retrieving Metadata with the Zip File.

RedirectWhitelistUrl

Represents a whitelisted URL that users can navigate to without being shown a warning message. This type extends the Metadata metadata type and inherits its `fullName` field.

File Suffix and Directory Location

RedirectWhitelistUrl components have the suffix `.redirectWhitelistUrl` and are stored in the `redirectWhitelistUrls` folder.

Version

RedirectWhitelistUrl components are available in API version 48.0 and later.

Fields

Field Name	Field Type	Description
`Url`	string	Required. The whitelisted URL.

Wildcard Support in the Manifest File

This metadata type supports the wildcard character `*` (asterisk) in the `package.xml` manifest file. For information about using the manifest file, see Deploying and Retrieving Metadata with the Zip File.

RecommendationStrategy

Represents a recommendation strategy. Recommendation strategies are applications, similar to data flows, that determine a set of recommendations to be delivered to the client through data retrieval, branching, and logic operations.

File Suffix and Directory Location

RecommendationStrategy components have the suffix `.recommendationStrategy` and are stored in the `recommendationStrategies` folder.

Version

RecommendationStrategy components are available in API version 45.0 and later.

Special Access Rules

Metadata access for the RecommendationStrategy type is backed by the ManageRecommendationStrategies user permission.

Fields

Field Name	Field Type	Description
`actionContext`	StrategyAction[]	An array of action contexts used by the strategy.
`contextRecordType`	string	The sObject type of the $record used by the flow.

Field Name	Field Type	Description
description	string	Description of the recommendation strategy.
filter	StrategyNodeFilter[]	An array of filter nodes.
if	StrategyNodeIf[]	An array of if nodes.
invocableAction	StrategyNodeInvocableAction[]	An array of Apex invocable action nodes. Available in API version 46.0 and later.
isTemplate	boolean	Indicates whether the recommendation strategy is a template (true) or not (false). When installed from managed packages, recommendation strategies can't be viewed or cloned by subscribers because of intellectual property (IP) protection. But when those recommendation strategies are templates, subscribers can open them in a builder, clone them, and customize the clones. The default value of this field is false. Available in API version 47.0 and later.
label	string	Required. Label for the flow.
map	StrategyNodeMap[]	An array of map nodes. Available in API version 46.0 and later.
mutuallyExclusive	StrategyNodeExclusive[]	An array of mutuallyExclusive nodes.
onBehalfOfExpression	string	Formula expression defining the intended target of the recommendations (in other words, the Contact associated with a Case). Mainly used for reaction tracking.
recommendationLimit	StrategyNodeRecommendationLimit[]	An array of recommendation limit nodes.
recommendationLoad	StrategyNodeRecommendationLoad[]	An array of recommendation load nodes.
sort	StrategyNodeSort[]	An array of sort nodes.
union	StrategyNodeUnion[]	An array of union nodes.

StrategyNodeBase

Base class for all strategy nodes. This is an abstract class.

Field Name	Field Type	Description
childNode	string	Array of child node names, in order of execution.
description	string	Description of the node.
label	string	Label of the node.
name	string	Required. Unique name of the node.

StrategyAction

Defines a call to an invocable action from the strategy. Results are used by decision elements in the strategy.

Field Name	Field Type	Description
action	string	Required. The name or id of the InvocableAction to execute.
argument	StrategyActionArg[]	List of strategy action arguments.
description	string	Description of the strategy.
label	string	Label for the strategy action.
name	string	Required. Unique name of the strategy action, which is referenced by decisioning elements in the strategy.
type	InvocableActionType (enumeration of type string)	Required. Type of the invocable action. See the enum InvocableActionType.

StrategyActionArg

Defines arguments passed to invocable actions associated with a strategy action.

Field Name	Field Type	Description
name	string	Required. Unique name for the parameter to pass to the invocable action.
value	string	Required. A Salesforce formula expression that is evaluated with the result being used as the parameter value for the Strategy Action.

StrategyNodeUnionBase

Base class for nodes that perform a union of their children. Union nodes combine the outputs of their children to form the input to themselves. StrategyNodeUnionBase extends StrategyNodeBase and inherits all of its fields. This is an abstract class.

Field Name	Field Type	Description
limit	int	Maximum number of results to output.

StrategyNodeFilter

Defines a filter element that filters recommendations. It extends StrategyNodeUnionBase and inherits all its fields.

Field Name	Field Type	Description
expression	string	Required. A formula expression that results in a boolean value when executed on each recommendation in the node's input. Inputs that result in true form the output, and inputs that result in false are excluded.

StrategyNodeIf

Selects specific children to execute and combines their results. Executes and returns results of children based on the array of child node expressions. Extends StrategyNodeUnionBase and inherits all of its fields.

Field Name	Field Type	Description
`childNodeExpression`	IfExpression[]	Array of if expressions.
`onlyFirstMatch`	boolean	If `true`, selects only the results from the matching child. If `false`, selects and combines results from all matching children. The default value is `false`.

IfExpression

Expression used by StrategyNodeIf.

Field Name	Field Type	Description
`childName`	string	Required. Name of child to match.
`expression`	string	Required. Formula expression returning `true` or `false`.

StrategyNodeInvocableAction

Defines an element that calls an Apex invocable action to generate or enhance a list of recommendations. It extends StrategyNodeUnionBase and inherits all its fields.

Field Name	Field Type	Description
`action`	string	Required. The name of the invocable action to execute.
`argument`	StrategyNodeInvocableActionArg[]	List of arguments that are passed to the invocable action.
`isGenerator`	boolean	Required. If `true`, the UI displays the Generate element. If `false`, the UI displays the Enhance element. Defaults to `false`.
`type`	InvocableActionType (enumeration of type string)	Required. Type of the invocable action. See the enum InvocableActionType. Valid value is `apex`.

StrategyNodeInvocableActionArg

Defines arguments passed to an Apex invocable action that generates or enhances a list of recommendations.

Field Name	Field Type	Description
`name`	string	Required. Unique name for the parameter to pass to the invocable action. The name must match a parameter that's defined in the invocable action.

Field Name	Field Type	Description
`value`	string	Required. A Salesforce formula expression that is evaluated with the result used as the parameter value for the action.

StrategyNodeRecommendationLimit

Filters out recommendations that have already been accepted or rejected. Extends StrategyNodeUnionBase and inherits all of its fields.

Field Name	Field Type	Description
`filterMode`	StrategyReactionType (enumeration of type string)	Available reactions to filter out. The valid values are: • `Accepted` • `Rejected`
`lookbackDuration`	int	Number of days to search back.
`maxRecommendationCount`	int	Maximum number of times recommendation has been accepted or rejected.

StrategyNodeRecommendationLoad

Retrieves Recommendation objects. Extends StrategyNodeUnionBase and inherits all of its fields.

Field Name	Field Type	Description
`condition`	RecommendationLoadCondition[]	Array of conditions specifying which recommendations to load.
`conditionLogic`	string	Logic to combine conditions, either AND or OR. All conditions are combined (not mixed). For example: `Cond1 AND Cond2 AND Cond3`.
`object`	string	Required. Specifies the API name of the sObject from which recommendations are loaded. For example, the field references `Account` or `MyCustomObject__c` and not a specific record of that object. Available in API version 48.0 and later.
`sortField`	StrategyNodeSortField	The field to sort on. Available in API version 48.0 and later.

RecommendationLoadCondition

Represents a condition used as part of the query constructed by StrategyNodeRecommendationLoad.

Field Name	Field Type	Description
`field`	string	Required. Any field from Recommendation BPO (SOAP) object.
`operator`	RecommendationConditionOperator (enumeration of type string)	Required. Valid values are: • `EQUALS`

Field Name	Field Type	Description
		• GREATER_THAN • GREATER_THAN_OR_EQUAL_TO • LESS_THAN • LESS_THAN_OR_EQUAL_TO • NOT_EQUALS • LIKE • STARTS_WITH • ENDS_WITH= • CONTAINS
value	RecommendationConditionValue	Required. Constant value to use in query.

RecommendationConditionValue

Represents a value used as part of a RecommendationCondition.

Field Name	Field Type	Description
type	RecommendationConditionValueType (enumeration of type string)	Required. Valid values are: • TEXT • NUMBER • BOOLEAN • DATE • DATE_TIME • TIME
value	string	Required. The constant value.

StrategyNodeSortField

Defines the field to sort on for StrategyNodeSort and StrategyNodeRecommendationLoad.

Field Name	Field Type	Description
name	string	Required. Name of the field to sort.
nullsFirst	boolean	If true, null values are sorted to the beginning of the list. Defaults to false.
order	SortOrder (enumeration of type string)	Order in which the list is sorted. Defaults to Asc. Valid values are: • Asc (ascending) • Desc (descending)

StrategyNodeSort

Sorts the recommendations. Extends StrategyNodeUnionBase and inherits all of its fields.

Field Name	Field Type	Description
`field`	StrategyNodeSortField	Required. Field to sort on.

StrategyNodeUnion

StrategyNodeUnion combines the output of all its child nodes. StrategyNodeUnion is a concrete implementation of StrategtNodeUnionBase and inherits all its fields.

StrategyNodeMap

Set recommendation fields with values. Extends StrategyNodeUnionBase and inherits all of its fields.

Field Name	Field Type	Description
`mapExpression`	MapExpression on page 775[]	List of MaxExpressions.

StrategyNodeExclusive

Returns results from the first child node that has results and no other. Extends StrategyNodeUnionBase and inherits all its fields.

MapExpression

Sets the value for a recommendation field used by the strategy.

Field Name	Field Type	Description
`expression`	string	Required. A formula expression that results in a valid value supported by the data type specified in the `type` field.
`name`	string	Required. Recommendation field name that the expression sets the value for.
`type`	string	Required. The data type of the value resulting from the value in the `expression` field. Valid values are: • `BOOLEAN` • `CURRENCY` • `DATE` • `DOUBLE` • `DATE_TIME` • `INTEGER` • `LONG`

Field Name	Field Type	Description
		- PERCENT - TEXT - TIME

Declarative Metadata Sample Definition

The following is an example of a RecommendationStrategy component that references the previous definition.

```
<?xml version="1.0" encoding="UTF-8"?>
<RecommendationStrategy xmlns="http://soap.sforce.com/2006/04/metadata">
    <contextRecordType>Asset</contextRecordType>
    <description>Hills Brothers Coffee strategy to handle machine down
incidents</description>
    <if>
        <childNode>IfNoEscaladeOrBetterSupport</childNode>
        <childNode>IfModel</childNode>
        <description>If Machine Down</description>
        <label>RootNode</label>
        <name>RootNode</name>
        <childNodeExpression>
            <childName>IfModel</childName>
            <expression>ISPICKVAL($Record.Status, &quot;OutOfOrder&quot;)</expression>
        </childNodeExpression>
        <childNodeExpression>
            <childName>IfNoEscaladeOrBetterSupport</childName>
            <expression>ISPICKVAL($Record.Status, &quot;OutOfOrder&quot;)</expression>
        </childNodeExpression>
        <onlyFirstMatch>false</onlyFirstMatch>
    </if>
    <if>
        <childNode>LoadEscalade</childNode>
        <description>If Customer does not have escalade support plan</description>
        <label>IfNoEscaladeOrBetterSupport</label>
        <name>IfNoEscaladeOrBetterSupport</name>
        <childNodeExpression>
            <childName>LoadEscalade</childName>
            <expression>NOT(ISPICKVAL($Record.Account.SLA__c, &quot;Gold&quot;) ||
ISPICKVAL($Record.Account.SLA__c, &quot;Platinum&quot;))</expression>
        </childNodeExpression>
        <onlyFirstMatch>false</onlyFirstMatch>
    </if>
    <if>
        <childNode>LoadMiniDiagnostic</childNode>
        <childNode>LoadMaxiDiagnostic</childNode>
        <description>If Machine Model switch node</description>
        <label>IfModel</label>
        <name>IfModel</name>
        <childNodeExpression>
            <childName>LoadMiniDiagnostic</childName>
          <expression>$Record.Product2.Name == &quot;Mini Coffee Roaster&quot;</expression>
        </childNodeExpression>
        <childNodeExpression>
            <childName>LoadMaxiDiagnostic</childName>
          <expression>$Record.Product2.Name == &quot;Maxi Coffee Roaster&quot;</expression>

        </childNodeExpression>
        <onlyFirstMatch>false</onlyFirstMatch>
    </if>
    <label>HillsBrothersCoffee</label>
    <recommendationLoad>
        <description>Load upgrade to escalade support plan</description>
        <label>LoadEscalade</label>
        <name>LoadEscalade</name>
        <condition>
            <field>Name</field>
            <operator>EQUALS</operator>
            <value>
                <type>TEXT</type>
                <value>Upgrade your Maintenance Package</value>
            </value>
        </condition>
        <conditionLogic>and</conditionLogic>
    </recommendationLoad>
    <recommendationLoad>
        <description>Load Mini Coffee Roaster Diagnostic Troubleshooting
proposition</description>
        <label>LoadMiniDiagnostic</label>
        <name>LoadMiniDiagnostic</name>
        <condition>
            <field>Name</field>
            <operator>EQUALS</operator>
            <value>
                <type>TEXT</type>
                <value>Mini Coffee Roaster Diagnostic Troubleshooting</value>
            </value>
        </condition>
        <conditionLogic>and</conditionLogic>
    </recommendationLoad>
    <recommendationLoad>
        <description>Load Maxi Coffee Roaster Diagnostic Troubleshooting
proposition</description>
        <label>LoadMaxiDiagnostic</label>
        <name>LoadMaxiDiagnostic</name>
        <condition>
            <field>Name</field>
            <operator>EQUALS</operator>
            <value>
                <type>TEXT</type>
                <value>Maxi Coffee Roaster Diagnostic Troubleshooting</value>
            </value>
        </condition>
        <conditionLogic>and</conditionLogic>
    </recommendationLoad>
    <union>
        <childNode>RootNode</childNode>
        <label>Output</label>
        <name>Output</name>
    </union>
    <invocableAction>
        <action>MyInvocableApexClass</action>
        <isGenerator>true</isGenerator>
        <type>apex</type>
        <argument>
            <name>MyNameParam</name>
            <value>$User.FirstName</value>
        </argument>
        <argument>
            <name>MyIdParam</name>
            <value>$Record.Id</value>
        </argument>
    </invocableAction>
    <map>
        <expression>
            <name>Name</name>
            <expression>'Hello' & $User.FirstName</expression>
            <type>TEXT</type>
        </expression>
        <expression>
            <name>MyDynamicField</name>
            <expression>Id == $Record.Id</expression>
            <type>BOOLEAN</type>
        </expression>
    </map>
</RecommendationStrategy>
```

Wildcard Support in the Manifest File

This metadata type supports the wildcard character `*` (asterisk) in the `package.xml` manifest file. For information about using the manifest file, see Deploying and Retrieving Metadata with the Zip File.

RecordActionDeployment

Represents configuration settings for the Actions & Recommendations component. For example, you can have a deployment that specifies which types of actions to display, default actions for channels, and the actions that users can add at runtime. If the component shows Next Best Action recommendations, the deployment configures which strategies to use and how recommendations appear. This type extends the Metadata metadata type and inherits its `fullName` field.

File Suffix and Directory Location

RecordActionDeployment values are stored in the *`developer_name`*`.deployment` file in the `recordActionDeployments` directory.

Note: We don't recommend programmatically changing the API name of a RecordActionDeployment.

Version

RecordActionDeployment is available in API version 45.0 and later.

Fields

Field Name	Field Type	Description
`channelConfigurations`	RecordActionDeploymentChannel	Specifies configuration settings for different channels in an Actions & Recommendations deployment.
`deploymentContexts`	RecordActionDeploymentContext	Specifies the object context for quick actions and Next Best Action strategies. Available in API version 46.0 and later.
`hasGuidedActions`	boolean	Specifies that the component shows standard actions; for example, flows and quick actions. Available in API version 46.0 and later.
`hasRecommendations`	boolean	Specifies that the component shows recommendations from a Next Best Action strategy. Available in API version 46.0 and later.
`masterLabel`	string	Required. Specifies the name of the deployment.
`recommendation`	RecordActionRecommendation	Specifies settings for how Next Best Action recommendations appear in the component. Available in API version 46.0 and later.
`selectableItems`	RecordActionSelectableItem	Specifies the actions that users can add at runtime.
`shouldLaunchActionOnReject`	boolean	Required. If `true`, launch the flow when the recommendation is rejected by the agent. Available in API version 48.0 and later.

RecordActionDefaultItem

Represents actions and attributes specified as channel defaults in a deployment.

Field Name	Field Type	Description
`action`	string	Required. Specifies the API name of an action. For example, the API name of a flow, such as `Verify_Information`.
`isMandatory`	boolean	Specifies whether the action is marked as mandatory. The default value is false.
`isUiRemoveHidden`	boolean	Specifies whether the remove option is hidden in the UI. The default value is false. If true, the UI hides the ability to remove the action from the list.
`pinned`	PinnedAction (enumeration of type string)	Required. Indicates whether the action is pinned to the `Top` or `Bottom`, or unpinned (`None`). The default value is `None`.
`position`	int	Required. Indicates the order of the action among all actions associated with this record.

Field Name	Field Type	Description
`type`	RecordActionType (enumeration of type string)	Required. The type of action that's associated with the record. Valid values are: • `Flow` • `QuickAction` (Available in API version 46.0 and later.)

RecordActionDeploymentChannel

Specifies channel-specific defaults to show in the Actions & Recommendations component. The component displays the channel defaults when the list is otherwise empty.

Field Name	Field Type	Description
`channel`	ChannelSource (enumeration of type string)	Required. Specifies the channel. Valid values are `Phone`, `Chat`, or `Default`.
`channelItems`	RecordActionDefaultItem	Specifies default actions for a channel and attributes for each action, such as whether the action is pinned to the list top or bottom or whether an action is considered mandatory.
`isAutopopEnabled`	boolean	Specifies whether the first action in the list is launched when the record page opens. If true, the first action is launched. The default value is false.

RecordActionDeploymentContext

Specifies an object that provides context for quick actions and Next Best Action strategies. When the component appears on this type of page, it includes object-specific quick actions and uses an object-specific strategy to filter recommendations. Available in API version 46.0 and later.

Note: We support a maximum of 10 objects that provide context within a deployment.

Field Name	Field Type	Description
`entityName`	string	Required. Specifies the API name of an object to use as context.
`recommendationStrategy`	string	Specifies the API name of a Next Best Action strategy that overrides the default strategy on this page. A strategy is a metadata type RecommendationStrategy.

RecordActionRecommendation

Specifies settings to display Next Best Action recommendations in the component. Available in API version 46.0 and later.

Field Name	Field Type	Description
`defaultStrategy`	string	Specifies the API name of the default Next Best Action strategy, which is a metadata type RecommendationStrategy.
`hasDescription`	boolean	Required. If `true`, display the description for the recommendation.

Field Name	Field Type	Description
hasImage	boolean	Required. If true, display the image for the recommendation.
hasRejectAction	boolean	Required. If true, display the label that the user clicks to reject the recommendation.
hasTitle	boolean	Required. If true, display the title for the recommendation.
maxDisplayRecommendations	int	Required. Specifies the maximum number of recommendations to display. Valid values are 1–4.

RecordActionSelectableItem

Represents the set of actions available for users to add to the component at runtime.

Field Name	Field Type	Description
action	string	Required. Specifies the API name of an action. For example, the API name of a flow, such as Verify_Information.
type	RecordActionType (enumeration of type string)	Required. The type of action that's associated with the record. Valid values are: • Flow • QuickAction (Available in API version 46.0 and later.)

Wildcard Support in the Manifest File

This metadata type supports the wildcard character * (asterisk) in the package.xml manifest file. For information about using the manifest file, see Deploying and Retrieving Metadata with the Zip File.

Declarative Metadata Sample Definition

The following is a sample of a recordActionDeployment file.

```
<RecordActionDeployment xmlns="http://soap.sforce.com/2006/04/metadata">
    <channelConfigurations>
        <channel>Phone</channel>
        <channelItems>
            <action>Sample_Flow</action>
            <isMandatory>false</isMandatory>
            <isUiRemoveHidden>false</isUiRemoveHidden>
            <position>1</position>
            <pinned>Top</pinned>
            <type>Flow</type>
        </channelItems>
        <channelItems>
            <action>Another_Sample_Flow</action>
            <isMandatory>false</isMandatory>
            <isUiRemoveHidden>true</isUiRemoveHidden>
            <position>2</position>
            <pinned>Top</pinned>
            <type>Flow</type>
        </channelItems>
        <isAutopopEnabled>true</isAutopopEnabled>
    </channelConfigurations>
    <masterLabel>Sample Deployment</masterLabel>
    <selectableItems>
        <action>Sample_Flow</action>
        <type>Flow</type>
    </selectableItems>
    <selectableItems>
        <action>Sample_Flow_2</action>
        <type>Flow</type>
    </selectableItems>
    <hasGuidedActions>true</hasGuidedActions>
    <hasRecommendations>true</hasRecommendations>
    <recommendation>
        <defaultStrategy>Sample_Global_Strategy</defaultStrategy>
        <maxDisplayRecommendations>4</maxDisplayRecommendations>
        <hasImage>true</hasImage>
        <hasDescription>true</hasDescription>
        <hasRejectAction>true</hasRejectAction>
        <hasTitle>true</hasTitle>
    </recommendation>
    <deploymentContexts>
        <entityName>Case</entityName>
        <recommendationStrategy>Sample_Case_Strategy</recommendationStrategy>
    </deploymentContexts>
    <deploymentContexts>
        <entityName>Account</entityName>
        <recommendationStrategy>Sample_Acc_Strategy</recommendationStrategy>
    </deploymentContexts>
</RecordActionDeployment>
```

The following is an example `package.xml` that references the previous definition.

```
<?xml version="1.0" encoding="UTF-8"?>
<Package xmlns="http://soap.sforce.com/2006/04/metadata">
    <fullName>SecondTest</fullName>
    <types>
        <members>Sample_Flow</members>
        <members>Another_Sample_Flow</members>
        <members>Sample_Flow_2</members>
        <name>Flow</name>
    </types>
    <types>
        <members>SampleDeployment</members>
        <name>RecordActionDeployment</name>
    </types>
    <version>45.0</version>
</Package>
```

SEE ALSO:

RecommendationStrategy

RemoteSiteSetting

Represents a remote site setting. Before any Visualforce page, Apex callout, or JavaScript code using XmlHttpRequest in an s-control or custom button can call an external site, that site must be registered in the Remote Site Settings page, or the call fails.

RemoteSiteSetting extends the Metadata metadata type and inherits its `fullName` field.

Declarative Metadata File Suffix and Directory Location

RemoteSiteSetting components are stored in the `remoteSiteSettings` directory of the corresponding package directory. The file name matches the unique name of the remote site setting, and the extension is `.remoteSite`.

Version

RemoteSiteSetting components are available in API version 19.0 and later.

Fields

Field	Field Type	Description
`description`	string	The description explaining what this remote site setting is used for.
`disableProtocolSecurity`	boolean	Required. Indicates whether code within Salesforce can access the remote site regardless of whether the user's connection is over HTTP or HTTPS (`true`) or not (`false`). When `true`, code within Salesforce can pass data from an HTTPS session to an HTTP session, and vice versa. Warning: Only set to `true` if you understand the security implications.
`fullName`	string	The name can only contain characters, letters, and the underscore (_) character, must start with a letter, and cannot end with an underscore or contain two consecutive underscore characters. Inherited from the Metadata component, this field is not defined in the WSDL for this component. It must be specified when creating, updating, or deleting. See `create()` to see an example of this field specified for a call.
`isActive`	boolean	Required. Indicates if the remote site setting is active (`true`) or not (`false`).
`url`	string	Required. The URL for the remote site.

Declarative Metadata Sample Definition

A sample XML definition of a remote site setting is shown below.

```
<?xml version="1.0" encoding="UTF-8"?>
<RemoteSiteSetting xmlns="http://soap.sforce.com/2006/04/metadata">
    <description>Used for Apex callout to mapping web service</description>
    <disableProtocolSecurity>false</disableProtocolSecurity>
    <isActive>true</isActive>
    <url>https://www.maptestsite.net/mapping1</url>
</RemoteSiteSetting>
```

Report

Represents a custom report. This metadata type only supports custom reports; standard reports are not supported.

This type extends the Metadata metadata type and inherits its `fullName` field.

Declarative Metadata File Suffix and Directory Location

Reports are stored in the `reports` directory of the corresponding package directory. The file name matches the report title and the extension is `.report`.

Retrieving Reports

You can't use the wildcard (*) symbol with reports in `package.xml`. To retrieve the list of reports for populating `package.xml` with explicit names, call `listMetadata()` and pass in `ReportFolder` as the type. Note that ReportFolder is not returned as a type in `describeMetadata()`. Report is returned from `describeMetadata()` with an associated attribute of `inFolder` set to true. If that attribute is set to true, you can construct the type by using the component name with the word Folder, such as ReportFolder.

The following example shows folders in `package.xml`:

```
<?xml version="1.0" encoding="UTF-8"?>
<Package xmlns="http://soap.sforce.com/2006/04/metadata">
    <types>
        <members>MyDBFolder/MyDBName</members>
        <name>Dashboard</name>
    </types>
    <types>
        <members>MyDocumentFolder/MyDocumentName</members>
        <name>Document</name>
    </types>
    <types>
        <members>unfiled$public/MarketingProductInquiryResponse</members>
        <members>unfiled$public/SalesNewCustomerEmail</members>
        <name>EmailTemplate</name>
    </types>
    <types>
        <members>MyReportFolder/MyReportName</members>
        <name>Report</name>
    </types>
    <version>48.0</version>
</Package>
```

Version

Report components are available in API version 14.0 and later.

Fields

The following information assumes that you are familiar with creating and running reports. For more information on these fields, see "Create a Report" in the Salesforce online help.

Field	Field Type	Description
aggregates	ReportAggregate[]	List that defines custom summary formulas for summary, matrix, and joined reports.
block	Report[]	Represents each block in a joined report where every block can be of a different report type.
blockInfo	ReportBlockInfo	Defines attributes for each block in a joined report.
buckets	ReportBucketField[]	Defines a bucket field to be used in the report. This field is available in API version 24.0 and later.
chart	ReportChart	Defines a chart for summary and matrix reports .
colorRanges	ReportColorRange[]	List that specifies conditional highlighting for report summary data. Salesforce Classic only.
columns	ReportColumn[]	List that specifies the fields displayed in the report. Fields appear in the report in the same order as they appear in the Metadata API file.
crossFilters	ReportCrossFilter[]	Defines a cross filter's object, related object, and condition (WITH or WITHOUT). This field is available in API version 48.0 and later.
currency	CurrencyIsoCode (enumeration of type string)	When using multiple currencies, some reports allow you to display converted amounts by selecting the appropriate column to display. For example, in opportunity reports, you can include the Amount (converted) column on the report. This field is an enumeration of type string that defines the currency in which to display converted amounts. Valid values: Must be one of the valid alphabetic, three-letter currency ISO codes defined by the ISO 4217 standard, such as USD, GBP, or JPY.

Field	Field Type	Description
`dataCategoryFilters`	string	Specifies a filter according to data category.
`description`	string	Specifies a general description, which is displayed with the report name. Maximum characters: 255 characters.
`division`	string	If your organization uses divisions to segment data and you have the “Affected by Divisions” permission, records in the report must match this division. This field is available in API version 17.0 and later.

`filter`	ReportFilter	Limits report results to records with specific data. For example, you can limit report results to opportunities for which the amount is greater than $1,000:
		`<filter>`
		` <criteriaItems>`
		`  <column>AMOUNT</column>`
		
		`<operator>greaterThan</operator>`
		
		`  <value>1000</value>`
		` </criteriaItems>`
		`</filter>`

`folderName`	string	Name of the folder that houses the report. This field is available in API version 35.0 and later.
`format`	ReportFormat (enumeration of type string)	Defines the report format. For example, `Tabular` for a simple data list without subtotals.
`formattingRules`	ReportFormattingRule[] (enumeration of type string)	List that specifies conditional highlighting for report data. Lightning Experience only.
`groupingsAcross`	ReportGrouping[]	List that defines the fields by which you want to group and subtotal data across a matrix report (row headings). When grouping by a date field, you can further group the data by a specific time period such as days, weeks, or months. Maximum: 2 fields.
`groupingsDown`	ReportGrouping[]	For Summary and Matrix reports: List that defines the fields by which you want to group and subtotal. For summary reports, choosing more than one sort field allows you to subsort your data. For matrix reports, specifies summary fields for column headings. When grouping by a date

Field	Field Type	Description
		field, you can further group the data by a specific time period such as days, weeks, or months. Maximum for matrix reports: 2. Maximum for summary reports: 3
`historicalSelector`	ReportHistoricalSelector	Defines a date range for which historical trend reporting data is to be captured. Default is "Any Historical Date." Available in API version 29.0 and later.
`name`	string	Required. The report name. For example, `Opportunity Pipeline`
`numSubscriptions`	int	Indicates whether a user has subscribed to this report Lightning Experience (1) or not (0). Tied to user context. This field is available in API version 38.0 and later.
`params`	ReportParam[]	List that specifies settings specific to each report type, in particular options that let you filter a report to obtain useful subsets. For example, the Activities report type lets you specify whether you want to see open or closed activities or both and whether you want to see tasks or events or both. Valid values depend on the report type.
`reportCustomDetailFormula`	CustomDetailFormulas	Allows you to apply row-level formulas to reports.
`reportType`	string	Required. Defines the type of data in the report. For example, `Opportunity` to create a report of opportunities data.
`roleHierarchyFilter`	string	The role name for a report drill down. Some reports, such as opportunity and activity reports, display Hierarchy links that allow you to drill down to different data sets based on the role hierarchy. This field is available in API version 17.0 and later.
`rowLimit`	int	Defines the maximum number of rows that can be returned for the report.
`scope`	string	Defines the scope of data on which you run the report. For example, whether you want to run the report against all opportunities, opportunities you own, or opportunities your team owns. Valid values depend on the

Field	Field Type	Description
		reportType. For example, for Accounts reports: - MyAccounts - MyTeamsAccounts - AllAccounts
showCurrentDate	boolean	Can be set to true for historical trending reports in matrix format. Available in API version 29.0 and later.
showDetails	boolean	false shows a collapsed view of the report with only the headings, subtotals, and total. Default: true
showGrandTotal	boolean	true displays the calculated total for the full report.
showSubTotals	boolean	true displays the calculated subtotals for sections of the report.
sortColumn	string	Specifies the field on which to sort data in the report. Use sortOrder to specify the sort order.
sortOrder	SortOrder (enumeration of type string)	Specifies the sort order. Use sortColumn to specify the field on which to sort.
territoryHierarchyFilter	string	The territory name for a report drill down. If your organization uses territory management, some reports display Hierarchy links that allow you to drill down to different data sets based on the territory hierarchy. This field is available in API version 17.0 and later.
timeFrameFilter	ReportTimeFrameFilter	Limits report results to records within a specified time frame.
userFilter	string	The user name for a report drill down. Some reports, such as opportunity and activity reports, display Hierarchy links that allow you to drill down to different data sets based on the user hierarchy. This field is available in API version 17.0 and later.

ReportAggregate

ReportAggregate defines custom summary formulas on summary, matrix, and joined reports. For more information on these fields, see "Add a Summary Formula Column to a Report" in the Salesforce online help.

Field	Field Type	Description
`acrossGroupingContext`	string	Defines the row grouping level at which you want your custom summary formula to be displayed. This is a new field in API version 15.0.
`calculatedFormula`	string	Required. The custom summary formula. For example, `AMOUNT:SUM + OPP_QUANTITY:SUM`
`datatype`	ReportAggregateDatatype (enumeration of type string)	Required. Specifies the data type for formatting and display of the custom summary formula results.
`description`	string	The custom summary formula description. Maximum: 255 characters.
`developerName`	string	Required. The internal development name of the custom summary formula, for example, `FORMULA1`. This is used to reference custom summary formulas from other report components, including conditional highlighting.
`downGroupingContext`	string	Defines the column grouping level at which you want your custom summary formula to be displayed. This field is available in API version 15.0 and later.
`isActive`	booleanblo	Required. `true` displays the formula result in the report. `false` does not display the result in the report.
`isCrossBlock`	boolean	Determines whether the custom summary formula is a cross-block formula, which is available with joined reports. `true` indicates a cross-block custom summary formula. `false` indicates a standard custom summary formula. This field is available in API version 25.0 and later.
`masterLabel`	string	Required. The custom summary formula label (name).
`reportType`	string	Required for joined reports. Specifies the `reportType` of the blocks to which the `aggregate` can be added.
`scale`	int	The formula result is calculated to the specified number of decimal places. Valid values `0` through `18`.

ReportBlockInfo

ReportBlockInfo defines blocks in a joined report.

Field	Field Type	Description
`aggregateReferences`	ReportAggregateReference[]	Lists the `aggregates` that represent the custom summary formulas used in a joined report block.
`blockId`	string	Required. `blockId` is used in cross-block custom summary formulas and joined report charts to identify the block containing each summary field. `blockId` is assigned automatically. Valid values are B1 through B5. This field is available in API version 25.0 and later.
`joinTable`	string	Required. Refers to the entity used to join blocks in a joined report. The entity provides a list of fields that are available for globally grouping across the blocks.

ReportAggregateReference

ReportAggregateReference defines the developer name used for custom summary formulas in joined reports.

Field	Field Type	Description
`aggregate`	string	Required. The `developerName` of the ReportAggregate, which specifies the custom summary formula used in a block of a joined report.

ReportBucketField

ReportBucketField defines a bucket to be used in the report.

Field	Field Type	Description
`bucketType`	ReportBucketFieldType (enumeration of type string)	Required. Specifies the type of bucket. Valid values: • text • number • picklist
`developerName`	string	Required. A unique name used as the `<field>` value to display a bucket field in the column list and other report components, including sort, filter, list, group, and chart. Must be of the format `BucketField_`***`name`***. For example, `BucketField_BusinessSize`.
`masterLabel`	string	Required. The bucket field label. Maximum 40 characters. Any line breaks, tabs, or multiple spaces at the beginning or end of the label are removed. Any of these characters within the label are reduced to a single space.

Field	Field Type	Description
`nullTreatment`	ReportBucketFieldNullTreatment (enumeration of type string)	For numeric bucket fields only. Specifies whether empty values are treated as zeros (`z`) or not (`n`).
`otherBucketLabel`	string	The label of the container for unbucketed values.
`sourceColumnName`	string	Required. The source field that the bucket is applied to. For example, `SALES` or `INDUSTRY`.
`values`	ReportBucketFieldValue (enumeration of type string)	Defines one bucket value used in the bucket field. Note: While this name is plural, it represents a single bucket. In typical use, a bucket field contains multiple buckets.

ReportBucketFieldValue

ReportBucketFieldValue defines a bucket value used in the bucket field.

Field	Field Type	Description
`sourceValues`	ReportBucketFieldSourceValue (enumeration of type string)	The value of a bucket in the bucket field. Valid values: • `sourceValue`—Used for picklist and text bucket fields. For picklists, describes the picklist item in the bucket. For example, the sourceValue of a bucket on `TYPE` could be `Customer`. For text, the full string for the item in the bucket. For example, the sourceValue of a bucket on `ADDRESS_STATE1` could be `NY`. • `from`—Used only on numeric bucket fields. A non-inclusive lower bound for a numeric bucket range. This value must be a number. • `to`—Used only on numeric bucket fields. The inclusive upper bound for a numeric bucket range. This value must be a number. In numeric buckets, the first value must only have `to` and last value must only have `from`. All other values must have both `to` and `from`.
`value`	string	Required. The name of a specific bucket value within the bucket field.

ReportGrouping

ReportGrouping defines how to group, subtotal, and sort data for summary, matrix, and joined reports.

Field	Field Type	Description
`aggregateType`	ReportAggrType (enumeration of type string)	The type of aggregate value to sort by. Valid values are: • `Sum` • `Average` • `Maximum` • `Minimum` • `RowCount` • `Unique`
`dateGranularity`	UserDateGranularity (enumeration of type string)	When grouping by a date field, the time period by which to group.
`field`	string	Required. The field by which you want to summarize data. For example, `CAMPAIGN_SOURCE`
`sortByName`	string	The API name of the column, aggregate or custom summary field used to order the grouping.
`sortOrder`	SortOrder	Required. Whether to sort data in ascending or descending alphabetical and numerical order.
`sortType`	ReportSortType (enumeration of type string)	Indicates if the grouping is sorted by a column, aggregate or custom summary field. Valid values are: • `Column` • `Aggregate` • `CustomSummaryFormula`

ReportHistoricalSelector

ReportHistoricalSelector defines a date range for historical data.

Field	Field Type	Description
`snapshot`	string	Represents the date value to apply a historical filter, either relative (in the format `N_DAYS_AGO:2`) or absolute (in the format `yyyy-MM-dd`). If unspecified, it's assumed that the filter will be applied to all the columns the user sees. Available in API version 29.0 and later.

CustomDetailFormulas

CustomDetailFormulas defines row-level formulas for reports.

Field	Field Type	Description
calculatedFormula	string	Required. The custom formula. For example, AMOUNT:SUM + OPP_QUANTITY:SUM
datatype	ReportCustomDetailFormulaDatatype (enumeration of type string)	Required. Specifies the data type for formatting and display of the formula results.
description	string	The formula description. Maximum: 255 characters.
developerName	string	Required. The internal development name of the formula, for example, FORMULA1. This is used to reference custom formulas from other report components, including conditional highlighting.
label	string	Required. The name that identifies this formula.
scale	int	The formula result is calculated to the specified number of decimal places. Valid values 0 through 18.

ReportCustomDetailFormulaDatatype

An enumeration of type string that specifies the data type for formatting and display of row-level formula results. Valid values:

Enumeration Value
Double
DateOnly
DateTime
Text

SortOrder

An enumeration of type string that defines the order in which data is sorted in the report fields. Valid values:

Field	Description
Asc	Sorts data in ascending alphabetical and numerical order.
Desc	Sorts data in descending alphabetical and numerical order.

UserDateGranularity

An enumeration of type string that defines the time period by which to group data. Valid values:

Enumeration Value	Description
None	No grouping by date

Enumeration Value	Description
`Day`	By day
`Week`	By week
`Month`	By month
`Quarter`	By quarter
`Year`	By year
`FiscalQuarter`	By fiscal quarter. You can set the fiscal year for your organization. See "Set the Fiscal Year" in the Salesforce online help.
`FiscalYear`	By fiscal year
`MonthInYear`	By calendar month in year
`DayInMonth`	By calendar day in month
`FiscalPeriod`	When custom fiscal years are enabled: By fiscal period
`FiscalWeek`	When custom fiscal years are enabled: By fiscal week

ReportSummaryType

An enumeration of type string that defines how report fields are summarized. Valid values:

Enumeration Value	Description
`Sum`	Total
`Average`	Average
`Maximum`	Largest value
`Minimum`	Smallest value
`Unique`	Unique values
`None`	The field is not summarized.

ReportColorRange

ReportColorRange defines conditional highlighting for report summary data.

Field	Field Type	Description
`aggregate`	ReportSummaryType (enumeration of type string)	Required. Defines how the field specified in `columnName` is summarized. For example, `Sum`.
`columnName`	string	Required. Specifies the field whose value ranges are represented by background colors.

Field	Field Type	Description
`highBreakpoint`	double	Required. Specifies the number that separates the mid color from the high color.
`highColor`	string	Required. Specifies the color (in HTML format) to represent data that falls into the high number range. This color spans from the `highBreakpoint` value.
`lowBreakpoint`	double	Required. Specifies the number that separates the low color from the mid color.
`lowColor`	string	Required. Specifies a color (in HTML format) to represent data that falls into the low value range, below the `lowBreakpoint` value.
`midColor`	string	Required. Specifies a color (in HTML format) to represent data that falls into the mid value range.

ReportColumn

ReportColumn defines how fields (columns) are displayed in the report.

Field	Field Type	Description
`aggregateTypes`	ReportSummaryType[] (enumeration of type string)	List that defines if and how each report field is summarized.
`field`	string	Required. The field name. For example, `AGE` or `OPPORTUNITY_NAME`
`reverseColors`	boolean	In historical trend reports, displays greater Date values as green and greater Amount values as red, reversing the default colors. Available in API version 29.0 and later.
`showChanges`	boolean	In historical trend reports, adds a column displaying the difference between current and historical Date and Amount values. Available in API version 29.0 and later.

ReportFilter

ReportFilter limits the report results by filtering data on specified fields.

Field	Field Type	Description
`booleanFilter`	string	Specifies filter logic conditions.

Field	Field Type	Description

`criteriaItems`	ReportFilterItem	The criteria by which you want to filter report data, either by comparing historical values or by applying a date range.
		`<criteriaItems>`
		`   criteriaItems ReportFilterItem`
		`<column>Opportunity.Opportunity__hd$Amount__hst</column>`
		`   <columnToColumn>false</columnToColumn>`
		`   <operator>equals</operator>`
		`   <snapshot>N_DAYS_AGO:90</snapshot>`
		`   <value>100</value>`
		`</criteriaItems>`

`language`	Language (enumeration of type string)	The language used when a report filters against a picklist value using the operators `contains` or `startsWith`. For a list of valid language values, see Language.

ReportFilterItem

ReportFilterItem limits the report results by filtering data on specified fields.

Field	Field Type	Description
`column`	string	Required. The field on which you want to filter data. For example, `AMOUNT`
`columnToColumn`	boolean	Indicates whether the filter is a column-to-column (field-to-field) filter. Available in API version 29.0 and later for historical trending reports. Available in API version 48.0 and later for general reports.
`isUnlocked`	boolean	Optional. Indicates whether the report filter is unlocked (`true`) or locked (`false`). You can edit unlocked filters on the report run page in Lightning Experience. If unspecified, the default value is `false`. Available in API version 38.0 and later.
`operator`	FilterOperation (enumeration of type string)	Required. An enumeration of type string that defines the operator used to filter the data, for example, `greaterThan`. For valid values, see FilterOperation.
`snapshot`	string	Represents the date value, either relative (in the format `N_DAYS_AGO:2`) or absolute (in the format `yyyy-MM-dd`). Available in API version 29.0 and later.

Field	Field Type	Description
value	string	The value by which you want to filter the data, for example, 1000. Note that the Metadata API filter condition values do not always match those that you enter in the report wizard. For example, in the Metadata API dates are always converted to the US date format and values entered in a non-US English language may be converted to a standard US English equivalent.

ReportFormat

An enumeration of type string that defines the report format. Valid values:

Enumeration Value	Description
Matrix	Summarizes data in a grid. Use to compare related totals.
Summary	Lists, sorts, and subtotals data.
Tabular	Lists data with no sorting or subtotals.
Joined	Joins data from different report types storing each report's data in its own block.

ReportFormattingRule

Defines conditional highlighting for report summary data. You can specify up to 5 formatting rules per report.

Field	Field Type	Description
aggregate	ReportFormattingSummaryType (enumeration of type string)	Defines how the field specified in columnName is summarized. For example, Sum.
columnName	string	Required. Specifies the field whose value ranges are represented by colors.
values	ReportFormattingRuleValue (enumeration of type string)	Required. Specifies the background colors and associated ranges for formatted data values.

ReportFormattingSummaryType

An enumeration of type string that defines how report fields are summarized. Valid values:

Enumeration Value	Description
Sum	Total
Average	Average
Maximum	Largest value
Minimum	Smallest value

Enumeration Value	Description
`Unique`	Unique values

ReportFormattingRuleValue

Specifies the background colors and associated ranges for formatted data values. You can specify up to 3 background colors and 0 to 3 range upper bounds. Valid values:

<table>
<tr><th>Field</th><th>Field Type</th><th>Description</th></tr>
<tr><td><code>backgroundColor</code></td><td>string</td><td>(Required) Specifies a highlighting color for the field in <code>columnName</code>. Must be a valid hex color string such as #54C254. At least one color is required. You can optionally specify a different color for up to 3 ranges as determined by <code>rangeUpperBound</code>. If you don’t specify a color for a particular range, the background is transparent.</td></tr>
<tr><td><code>rangeUpperBound</code></td><td>double</td><td>Delineates a range to which a background color applies. If you don’t specify an upper bound for a particular range, the bound is assumed to be plus infinity. The following example sets the background color for the Sales column to #B50E03 for aggregate sales less than or equal to 100, sets no background for sales from 100 to 1000, and sets the background color to #006714 for sales greater than 1000.

```
<formattingRules>
 <aggregate>Sum</aggregate>
 <columnName>Sales</columnName>
        <values>

<backgroundColor>#B50E03</backgroundColor>


<rangeUpperBound>100.0</rangeUpperBound>
        </values>
        <values>

<rangeUpperBound>1000.0</rangeUpperBound>

        </values>
        <values>

<backgroundColor>#006714</backgroundColor>

        </values>
</formattingRules>
```

</td></tr>
</table>

ReportParam

ReportParam represents settings specific to a report type, especially options that let you filter a report to certain useful subsets.

Field	Field Type	Description
`name`	string	Required. Specifies a specific `reportType` setting.
`value`	string	Required. The setting value.

ReportAggregateDatatype

An enumeration of type string that specifies the data type for formatting and display of custom summary formula results. Valid values:

Enumeration Value
`currency`
`number`
`percent`

ReportChart

ReportChart represents charts on summary, matrix, and joined reports.

Field	Field Type	Description
`backgroundColor1`	string	Specifies the beginning color (in HTML format) for a gradient color background.
`backgroundColor2`	string	Specifies the end color (in HTML format) for a gradient color background.
`backgroundFadeDir`	ChartBackgroundDirection (enumeration of type string)	Specifies the direction for a gradient color background. Use with `backgroundColor1` to specify the beginning color and `backgroundColor2` to specify the end color for the gradient design. Use white for both if you do not want a background design. Valid values: • `Diagonal` • `LeftToRight` • `TopToBottom`
`chartSummaries`	ChartSummary[]	Specifies the summaries you want to use for the chart. Invalid summaries are ignored without notification. If there are no valid summaries, RowCount is used by default for the axis value. This field is available in API version 17.0 and later.
`chartType`	ChartType (enumeration of type string)	Required. Specifies the chart type. Available chart types depend on the report type.

Field	Field Type	Description
enableHoverLabels	boolean	Specifies whether to display values, labels, and percentages when hovering over charts. Hover details depend on chart type. Percentages apply to pie, donut, and funnel charts only. This field is available in API version 17.0 and later.
expandOthers	boolean	Specifies whether to combine all groups less than or equal to 3% of the total into a single 'Others' wedge or segment. This only applies to pie, donut, and funnel charts. Set to `true` to show all values individually on the chart; set to `false` to combine small groups into 'Others.' This field is available in API version 17.0 and later.
groupingColumn	string	Specifies the field by which to group data. This data is displayed on the X-axis for vertical column charts and on the Y-axis for horizontal bar charts.
legendPosition	ChartLegendPosition (enumeration of type string)	Required. The location of the legend with respect to the chart. The valid values are: • `Bottom` • `OnChart` • `Right`
location	ChartPosition (enumeration of type string)	Required. Specifies whether the chart is displayed at the top or bottom of the report.
secondaryGroupingColumn	string	For grouped chart types: Specifies the field by which to group the data.
showAxisLabels	boolean	For bar and line charts: Specifies whether the chart displays names for each axis.
showPercentage	boolean	Indicates if percentages are displayed for wedges and segments of pie, donut, and funnel charts, as well as for gauges (`true`), or not (`false`).
showTotal	boolean	Indicates if the total is displayed for donut charts and gauges (`true`), or not (`false`).
showValues	boolean	Indicates if the values of individual records or groups are displayed for charts (`true`), or not (`false`).
size	ReportChartSize (enumeration of type string)	Required. Specifies the chart size.
summaryAggregate	ReportSummaryType (enumeration of type string)	Defines how to summarize the chart data. For example, `Sum`. No longer supported in version API 17.0 and later. See `chartSummaries`.
summaryAxisManualRangeEnd	double	When specifying the axis range manually: Defines the ending value.

Field	Field Type	Description
summaryAxisManualRangeStart	double	When specifying the axis range manually: Defines the starting value.
summaryAxisRange	ChartRangeType (enumeration of type string)	Required. For bar, line, and column charts: Defines whether to specify the axis range manually or automatically.
summaryColumn	string	Required. Specifies the field by which to summarize the chart data. Typically this field is displayed on the Y-axis. No longer supported in version API 17.0 and later. See chartSummaries.
textColor	string	The color (in HTML format) of the chart text and labels.
textSize	int	The size of the chart text and labels. Valid values: • 8 • 9 • 10 • 12 • 14 • 18 • 24 • 36 The maximum size is 18. Larger values are shown at 18 points.
title	string	The chart title. Max 255 characters.
titleColor	string	The color (in HTML format) of the title text.
titleSize	int	The size of the title text. Valid values: • 8 • 9 • 10 • 12 • 14 • 18 • 24 • 36 The maximum size is 18. Larger values are shown at 18 points.

ChartType

An enumeration of type string that defines the chart type. For information on each of these chart types, see "Chart Types" in the Salesforce online help. Valid values:

Enumeration Value
`None`
`HorizontalBar`
`HorizontalBarGrouped`
`HorizontalBarStacked`
`HorizontalBarStackedTo100`
`VerticalColumn`
`VerticalColumnGrouped`
`VerticalColumnStacked`
`VerticalColumnStackedTo100`
`Line`
`LineGrouped`
`LineCumulative`
`LineCumulativeGrouped`
`Pie`
`Donut`
`Funnel`
`Scatter`
`ScatterGrouped`
`VerticalColumnLine`
`VerticalColumnGroupedLine`
`VerticalColumnStackedLine`
`Plugin` Reserved for future use. This value is available in API version 31.0 and later.

ChartPosition

An enumeration of type string that specifies the position of the chart in the report. Valid values:

Enumeration Value
`CHART_TOP`
`CHART_BOTTOM`

ChartSummary

ChartSummary defines how data in the chart is summarized. Valid values:

Field	Field Type	Description
aggregate	ReportSummaryType	Specifies the aggregation method—such as Sum, Average, Min, and Max—for the summary value. Use the column field to specify the summary value to use for the aggregation. You don't need to specify this field for RowCount or custom summary formulas.
axisBinding	ChartAxis	Specifies the axis or axes to use on the chart. Use the column field to specify the summary value to use for the axis.
column	string	Required. Specifies the summary field for the chart data. If all columns are invalid, RowCount is used by default for the axis value. For vertical column and horizontal bar combination charts, you can specify up to four values.

ChartAxis

An enumeration of type string that specifies the axis or axes to be used in charts. Valid values:

Enumeration Value	Description
x	The summary value to use for the X-axis of a scatter chart.
y	The Y-axis for the chart.
y2	The secondary Y-axis for vertical column combination charts with a line added.

ReportChartSize

An enumeration of type string that specifies the chart size. Valid values:

Enumeration Value
Tiny
Small
Medium
Large
Huge

ChartRangeType

An enumeration of type string that defines the report format. Valid values:

Enumeration Value
Auto
Manual

ReportTimeFrameFilter

ReportTimeFrameFilter represents the report time period.

Field	Field Type	Description
dateColumn	string	Required. The date field on which to filter data. For example, CLOSE_DATE
endDate	date	When interval is INTERVAL_CUSTOM, specifies the end of the custom time period.
interval	UserDateInterval (enumeration of type string)	Required. Specifies the period of time.
startDate	date	When interval is INTERVAL_CUSTOM, specifies the start of the custom time period.

ReportCrossFilter

ReportCrossFilter represents the cross filter functionality in reports.

Field	Field Type	Description
criteriaItems	ReportFilterItem	Represents the subfilters of a cross filter. There can be up to five subfilters. This field requires the following attributes. - Column - Operator - Value
operation	ObjectFilterOperator. Enumeration of type string	The action indicating whether to include or exclude an object. Valid values: with and without.
primaryTableColumn	string	The parent object used for the cross filter.
relatedTable	string	The child object used for the cross filter.
relatedTableJoinColumn	string	The field from the child object that is used to join the parent.

Declarative Metadata Sample Definition

A sample XML snippet using cross filters to build an Accounts report for cases where case status is not closed:

```
    <crossFilters>
        <criteriaItems>
            <column>Status</column>
            <operator>notequal</operator>
            <value>Closed</value>
        </criteriaItems>
        <operation>with</operation>
        <primaryTableColumn>ACCOUNT_ID</primaryTableColumn>
        <relatedTable>Case</relatedTable>
        <relatedTableJoinColumn>Account</relatedTableJoinColumn>
    </crossFilters>
```

Note: This sample was generated using the API version 23.0.

UserDateInterval

An enumeration of type string that defines the period of time. Valid values:

Enumeration Value	Description
`INTERVAL_CURRENT`	Current fiscal quarter
`INTERVAL_CURNEXT1`	Current and next fiscal quarters
`INTERVAL_CURPREV1`	Current and previous fiscal quarters
`INTERVAL_NEXT1`	Next fiscal quarter
`INTERVAL_PREV1`	Previous fiscal quarter
`INTERVAL_CURNEXT3`	Current and next three fiscal quarters
`INTERVAL_CURFY`	Current fiscal year
`INTERVAL_PREVFY`	Previous fiscal year
`INTERVAL_PREV2FY`	Previous two fiscal years
`INTERVAL_AGO2FY`	Two fiscal years ago
`INTERVAL_NEXTFY`	Next fiscal year
`INTERVAL_PREVCURFY`	Current and previous fiscal years
`INTERVAL_PREVCUR2FY`	Current and previous two fiscal years
`INTERVAL_CURNEXTFY`	Current and next fiscal year
`INTERVAL_CUSTOM`	A custom time period. Use `startDate` and `endDate` fields to specify the time period's start date and end date.
`INTERVAL_YESTERDAY`	Yesterday

Enumeration Value	Description
INTERVAL_TODAY	Today
INTERVAL_TOMORROW	Tomorrow
INTERVAL_LASTWEEK	Last calendar week
INTERVAL_THISWEEK	This calendar week
INTERVAL_NEXTWEEK	Next calendar week
INTERVAL_LASTMONTH	Last calendar month
INTERVAL_THISMONTH	This calendar month
INTERVAL_NEXTMONTH	Next calendar month
INTERVAL_LASTTHISMONTH	Current and previous calendar months
INTERVAL_THISNEXTMONTH	Current and next calendar months
INTERVAL_CURRENTQ	Current calendar quarter
INTERVAL_CURNEXTQ	Current and next calendar quarters
INTERVAL_CURPREVQ	Current and previous calendar quarters
INTERVAL_NEXTQ	Next calendar quarter
INTERVAL_PREVQ	Previous calendar quarter
INTERVAL_CURNEXT3Q	Current and next three calendar quarters
INTERVAL_CURY	Current calendar year
INTERVAL_PREVY	Previous calendar year
INTERVAL_PREV2Y	Previous two calendar years
INTERVAL_AGO2Y	Two calendar years ago
INTERVAL_NEXTY	Next calendar year
INTERVAL_PREVCURY	Current and previous calendar years
INTERVAL_PREVCUR2Y	Current and previous two calendar years
INTERVAL_CURNEXTY	Current and next calendar years
INTERVAL_LAST7	Last 7 days
INTERVAL_LAST30	Last 30 days
INTERVAL_LAST60	Last 60 days
INTERVAL_LAST90	Last 90 days
INTERVAL_LAST120	Last 120 days
INTERVAL_NEXT7	Next 7 days

Enumeration Value	Description
INTERVAL_NEXT30	Next 30 days
INTERVAL_NEXT60	Next 60 days
INTERVAL_NEXT90	Next 90 days
INTERVAL_NEXT120	Next 120 days
LAST_FISCALWEEK	When custom fiscal years are enabled: Last fiscal week
THIS_FISCALWEEK	When custom fiscal years are enabled: This fiscal week
NEXT_FISCALWEEK	When custom fiscal years are enabled: Next fiscal week
LAST_FISCALPERIOD	When custom fiscal years are enabled: Last fiscal period
THIS_FISCALPERIOD	When custom fiscal years are enabled: This fiscal period
NEXT_FISCALPERIOD	When custom fiscal years are enabled: Next fiscal period
LASTTHIS_FISCALPERIOD	When custom fiscal years are enabled: This fiscal period and last fiscal period
THISNEXT_FISCALPERIOD	When custom fiscal years are enabled: This fiscal period and next fiscal period
CURRENT_ENTITLEMENT_PERIOD	Current entitlement period
PREVIOUS_ENTITLEMENT_PERIOD	Previous entitlement period
PREVIOUS_TWO_ENTITLEMENT_PERIODS	Previous two entitlement periods
TWO_ENTITLEMENT_PERIODS_AGO	Two entitlement periods ago
CURRENT_AND_PREVIOUS_ENTITLEMENT_PERIOD	Current and previous entitlement period
CURRENT_AND_PREVIOUS_TWO_ENTITLEMENT_PERIODS	Current and previous two entitlement periods

Declarative Metadata Sample Definition

A sample XML report definition:

```
<?xml version="1.0" encoding="UTF-8"?>
<Report xmlns="http://soap.sforce.com/2006/04/metadata">
    <aggregates>
        <acrossGroupingContext>CRT_Object__c$Id</acrossGroupingContext>
        <calculatedFormula>PREVGROUPVAL(CRT_Object__c.Currency__c:AVG, CRT_Object__c.Id)
*
                PARENTGROUPVAL(CRT_Object__c.Number__c:MAX, CRT_Object__c.CreatedBy.Name,

                COLUMN_GRAND_SUMMARY)/RowCount</calculatedFormula>
        <datatype>number</datatype>
        <developerName>FORMULA1</developerName>
        <downGroupingContext>CRT_Object__c$CreatedBy</downGroupingContext>
        <isActive>true</isActive>
        <masterLabel>CurrCSF</masterLabel>
        <scale>2</scale>
    </aggregates>
    <aggregates>
        <acrossGroupingContext>CRT_Object__c$LastModifiedDate</acrossGroupingContext>
        <calculatedFormula>IF(RowCount&gt;10,
                BLANKVALUE(ROUND(PREVGROUPVAL(CRT_Object__c.Currency__c:SUM,
                CRT_Object__c.LastModifiedDate),3),
                PARENTGROUPVAL(CRT_Object__c.Number__c:SUM, ROW_GRAND_SUMMARY,
                CRT_Object__c.Id))  , 1000)</calculatedFormula>
        <datatype>number</datatype>
        <developerName>FORMULA2</developerName>
        <downGroupingContext>GRAND_SUMMARY</downGroupingContext>
        <isActive>true</isActive>
        <masterLabel>numCSF</masterLabel>
        <scale>2</scale>
    </aggregates>
    <buckets>
        <bucketType>number</bucketType>
        <developerName>BucketField_BusinessSize</developerName>
        <masterLabel>NumericBucket</masterLabel>
        <nullTreatment>z</nullTreatment>
        <sourceColumnName>SALES</sourceColumnName>
        <values>
            <sourceValues>
                <to>10000</to>
            </sourceValues>
            <value>low</value>
        </values>
        <values>
            <sourceValues>
                <from>10000</from>
                <to>25000</to>
            </sourceValues>
            <value>mid</value>
        </values>
        <values>
            <sourceValues>
                <from>25000</from>
            </sourceValues>
            <value>high</value>
        </values>
    </buckets>
    <buckets>
        <bucketType>text</bucketType>
        <developerName>BucketField_Region</developerName>
        <masterLabel>TextBucket</masterLabel>
        <nullTreatment>n</nullTreatment>
        <otherBucketLabel>Other</otherBucketLabel>
        <sourceColumnName>ADDRESS1_STATE</sourceColumnName>
        <values>
            <sourceValues>
                <sourceValue>CA</sourceValue>
            </sourceValues>
            <value>west</value>
        </values>
```

```
            <values>
                <sourceValues>
                    <sourceValue>NY</sourceValue>
                </sourceValues>
                <sourceValues>
                    <sourceValue>Ontario</sourceValue>
                </sourceValues>
                <value>east</value>
            </values>
        </buckets>
        <chart>
            <backgroundColor1>#FFFFFF</backgroundColor1>
            <backgroundColor2>#FFFFFF</backgroundColor2>
            <backgroundFadeDir>Diagonal</backgroundFadeDir>
            <chartSummaries>
                <axisBinding>y</axisBinding>
                <column>FORMULA1</column>
            </chartSummaries>
            <chartSummaries>
                <axisBinding>y</axisBinding>
                <column>FORMULA2</column>
            </chartSummaries>
            <chartSummaries>
                <aggregate>Maximum</aggregate>
                <axisBinding>y</axisBinding>
                <column>CRT_Object__c$Number__c</column>
            </chartSummaries>
            <chartSummaries>
                <axisBinding>y</axisBinding>
                <column>RowCount</column>
            </chartSummaries>
            <chartType>VerticalColumn</chartType>
            <groupingColumn>CRT_Object__c$LastModifiedDate</groupingColumn>
            <legendPosition>Right</legendPosition>
            <location>CHART_TOP</location>
            <size>Medium</size>
            <summaryAxisRange>Auto</summaryAxisRange>
            <textColor>#000000</textColor>
            <textSize>12</textSize>
            <titleColor>#000000</titleColor>
            <titleSize>18</titleSize>
        </chart>
        <columns>
            <field>CRT_Object__c$Name</field>
        </columns>
        <columns>
            <aggregateTypes>Average</aggregateTypes>
            <field>CRT_Object__c$Currency__c</field>
        </columns>
        <columns>
            <aggregateTypes>Maximum</aggregateTypes>
            <field>CRT_Object__c$Number__c</field>
        </columns>
        <columns>
            <field>BucketField__Region</field>
        </columns>
        <format>Matrix</format>
        <groupingsAcross>
            <dateGranularity>Day</dateGranularity>
            <field>CRT_Object__c$Id</field>
            <sortOrder>Asc</sortOrder>
        </groupingsAcross>
        <groupingsAcross>
            <dateGranularity>Year</dateGranularity>
            <field>CRT_Object__c$LastModifiedDate</field>
            <sortOrder>Asc</sortOrder>
        </groupingsAcross>
        <groupingsDown>
            <dateGranularity>Day</dateGranularity>
            <field>CRT_Object__c$CreatedBy</field>
            <sortOrder>Asc</sortOrder>
        </groupingsDown>
        <groupingsDown>
            <dateGranularity>Day</dateGranularity>
            <field>CRT_Object__c$Currency__c</field>
            <sortOrder>Desc</sortOrder>
        </groupingsDown>
        <name>CrtMMVC</name>
        <reportType>CRT1__c</reportType>
        <scope>organization</scope>
        <showDetails>false</showDetails>
        <timeFrameFilter>
            <dateColumn>CRT_Object__c$CreatedDate</dateColumn>
            <interval>INTERVAL_CUSTOM</interval>
        </timeFrameFilter>
    </Report>
```

Declarative Metadata Sample Definition for a Joined Report

A sample XML report definition:

```
<?xml version="1.0" encoding="UTF-8"?>
<Report xmlns="http://soap.sforce.com/2006/04/metadata">
<!-- This is a cross-block custom summary formula. Note that the calculated formula reference
 for a blocks reference uses the BlockId#Aggregate. -->
    <aggregates>
        <calculatedFormula>B1#AMOUNT:SUM+B2#EMPLOYEES:SUM</calculatedFormula>
        <datatype>number</datatype>
        <developerName>FORMULA</developerName>
        <isActive>true</isActive>
        <isCrossBlock>true</isCrossBlock>
        <masterLabel>Cross-Block CSF Example</masterLabel>
        <scale>2</scale>
    </aggregates>
<!-- This is a standard custom summary formula. Note that the calculated formula reference
 does not have block reference but just the aggregate name of the report type associated
(Opportunity).-->
    <aggregates>
```

```
        <calculatedFormula>AMOUNT:SUM</calculatedFormula>
        <developerName>FORMULA2</developerName>
        <isActive>true</isActive>
        <isCrossBlock>false</isCrossBlock>
        <masterLabel>Standard CSF Example</masterLabel>
        <reportType>Opportunity</reportType>
        <scale>2</scale>
    </aggregates>
    <block>
      <blockInfo>
<!-- This is how the block defines that the custom summary formula should be referenced.
In this example, it’s the in standard FORMULA 2 defined above. This block report has blockID
 B1.-->
        <aggregateReferences>
          <aggregate>FORMULA2</aggregate>
        </aggregateReference>
        <blockId>B1</blockId>
        <joinTable>a</joinTable>
      </blockInfo>
      <columns>
        <field>TYPE</field>
      </columns>
      <format>Summary</format>
      <name>Opportunities BLock 3</name>
      <params>
        <name>role_territory</name>
        <value>role</value>
      </params>
      <params>
        <name>terr</name>
        <value>all</value>
      </params>
      <params>
        <name>open</name>
        <value>all</value>
      </params>
      <params>
        <name>probability</name>
        <value>0</value>
      </params>
      <params>
        <name>co</name>
        <value>1</value>
      </params>
      <reportType>Opportunity</reportType>
      <scope>organization</scope>
      <timeFrameFilter>
        <dateColumn>CLOSE_DATE</dateColumn>
        <interval>INTERVAL_CUSTOM</interval>
      </timeFrameFilter>
    </block>
    <block>
      <blockInfo>
<!-- This is how the block defines that the custom summary formula should be referenced.
```

```
In this example, it’s the cross-block custom summary formula FORMULA 1 defined above. This
 block report has blockId B2.-->
        <aggregateReferences>
          <aggregate>FORMULA1</aggregate>
        </aggregateReferences>
        <blockId>B2</blockId>
        <joinTable>a</joinTable>
      </blockInfo>
      <columns>
        <field>USERS.NAME</field>
      </columns>
      <columns>
        <field>TYPE</field>
      </columns>
      <columns>
         <field>DUE_DATE</field>
      </columns>
      <columns>
        <field>LAST_UPDATE</field>
      </columns>
      <columns>
        <field>ADDRESS1_STATE</field>
      </columns>
      <format>Summary</format
      <name>Accounts block 5</name>
      <params>
        <name>terr</name>
        <value>all</value>
      </params>
      <params>
        <name>co</name>
        <value>1</value>
      </params>
      <reportType>AccountList</reportType>
      <scope>organization</scope>
      <timeFrameFilter>
        <dateColumn>CREATED_DATE</dateColumn>
        <interval>INTERVAL_CUSTOM</interval>
      </timeFrameFilter>
    </block>
    <blockInfo>
      <blockId xsi:nil="true"/>
      <joinTable>a</joinTable>
    </blockInfo>
<chart>
        <backgroundColor1>#FFFFFF</backgroundColor1>
        <backgroundColor2>#FFFFFF</backgroundColor2>
        <backgroundFadeDir>Diagonal</backgroundFadeDir>
        <chartSummaries>
            <axisBinding>y</axisBinding>
<!-- This is how chart aggregates are designed in multiblock. We're using RowCount from
Block 1.-->
            <column>B1#RowCount</column>
        </chartSummaries>
        <chartType>HorizontalBar</chartType>
        <enableHoverLabels>false</enableHoverLabels>
        <expandOthers>true</expandOthers>
        <groupingColumn>ACCOUNT_NAME</groupingColumn>
        <location>CHART_TOP</location>
        <showAxisLabels>true</showAxisLabels>
        <showPercentage>false</showPercentage>
        <showTotal>false</showTotal>
        <showValues>false</showValues>
        <size>Medium</size>
        <summaryAxisRange>Auto</summaryAxisRange>
        <textColor>#000000</textColor>
        <textSize>12</textSize>
        <titleColor>#000000</titleColor>
        <titleSize>18</titleSize>
    </chart>
    <format>MultiBlock</format>
    <groupingsDown>
        <dateGranularity>Day</dateGranularity>
        <field>ACCOUNT_NAME</field>
        <sortOrder>Asc</sortOrder>
    </groupingsDown>
    <name>mb_mbapi</name>
    <reportType>Opportunity</reportType>
    <showDetails>true</showDetails>
</Report>
```

Wildcard Support in the Manifest File

This metadata type doesn't support the wildcard character `*` (asterisk) in the `package.xml` manifest file. For information about using the manifest file, see Deploying and Retrieving Metadata with the Zip File.

SEE ALSO:

Dashboard

ReportType

Represents the metadata associated with a custom report type. Custom report types allow you to build a framework from which users can create and customize reports. For more information, see “Set Up a Custom Report Type” in the Salesforce online help.

This type extends the Metadata metadata type and inherits its `fullName` field.

Declarative Metadata File Suffix and Directory Location

The file suffix is `.reportType` for the custom report type definition. There is one file per custom report type. Report types are stored in the `reportTypes` directory of the corresponding package directory.

Version

Custom report types are available in API version 14.0 and later.

Fields

Field Name	Field Type	Description
`autogenerated`	boolean	Indicates that the report type was automatically generated when historical trending was enabled for an entity. Available in API version 29 and later.
`baseObject`	string	Required. The primary object for the custom report type, for example, Account. All objects, including custom and external objects, are supported. You cannot edit this field after initial creation. Support for external objects is available in API version 38.0 and later.
`category`	ReportTypeCategory (enumeration of type string)	Required. This field controls the category for the report. The valid values are: • `accounts` • `opportunities` • `forecasts` • `cases` • `leads` • `campaigns` • `activities` • `busop` • `products` • `admin` • `territory` • `territory2` (This value is available in API version 31.0 and later.) • `usage_entitlement` • `wdc` (This value is available in API version 29.0 and later.) • `calibration` (This value is available in API version 29.0 and later.) • `other` • `content` • `quotes` • `individual` (This value is available in API version 45.0 and later.)
`deployed`	boolean	Required. Indicates whether the report type is available to users (`true`) or whether it's still in development (`false`).
`description`	string	The description of the custom report type.

Field Name	Field Type	Description
fullName	string	The report type developer name used as a unique identifier for API access. The fullName can contain only underscores and alphanumeric characters. It must be unique, begin with a letter, not include spaces, not end with an underscore, and not contain two consecutive underscores.
join	ObjectRelationship	The object joined to the baseObject. For example, Contacts may be joined to the primary Accounts object.
label	string	Required. The report type label.
sections	ReportLayoutSection[]	The groups of columns available for the report type. Though columns are not strictly required, a report without columns is not very useful.

ObjectRelationship

ObjectRelationship represents a join to another object. For more information, see "Add Child Objects To Your Custom Report Type" in the Salesforce online help.

Field Name	Field Type	Description
join	ObjectRelationship	This field is a recursive reference that allows you to join more than two objects. A maximum of four objects can be joined in a custom report type. When more than two objects are joined, an inner join is not allowed if there has been an outer join earlier in the join sequence. The baseObject is first joined to the object specified in relationship; the resulting data set is then joined with any objects specified in this field.
outerJoin	boolean	Required. Indicates whether this is an outer join (true) or not (false). An outer join returns a row even if the joined table does not contain a matching value in the join column.
relationship	string	Required. The object joined to the primary object; for example, Contacts.

ReportLayoutSection

ReportLayoutSection represents a group of columns used in the custom report type.

Field Name	Field Type	Description
columns	ReportTypeColumn[]	The list of columns projected from the query, defined by this custom report type.
masterLabel	string	Required. The label for this group of columns in the report wizard.

ReportTypeColumn

ReportTypeColumn represents a column in the custom report type.

Field Name	Field Type	Description
`checkedByDefault`	boolean	Required. Indicates whether this column is selected be default (`true`) or not (`false`).
`displayNameOverride`	string	A customized column name, if desired.
`field`	string	Required. The field name associated with the report column.
`table`	string	Required. The table associated with the field; for example, Account.

Declarative Metadata Sample Definition

The definition of a custom report type is shown below. Account is joined to Contacts and the resulting data set is joined with Assets.

```
<?xml version="1.0" encoding="UTF-8"?>
<ReportType xmlns="http://soap.sforce.com/2006/04/metadata">
    <baseObject>Account</baseObject>
    <category>accounts</category>
    <deployed>true</deployed>
    <description>Account linked to Contacts and Assets</description>
    <join>
        <join>
            <outerJoin>false</outerJoin>
            <relationship>Assets</relationship>
        </join>
        <outerJoin>false</outerJoin>
        <relationship>Contacts</relationship>
    </join>
    <label>Account Contacts and Assets</label>
    <sections>
        <columns>
            <checkedByDefault>true</checkedByDefault>
            <field>obj_lookup__c.Id</field>
            <table>Account</table>
        </columns>
        <columns>
            <checkedByDefault>false</checkedByDefault>
            <field>obj_lookup__c.Name</field>
            <table>Account</table>
        </columns>
        <columns>
            <checkedByDefault>false</checkedByDefault>
            <field>Opportunity__c.Amount</field>
            <table>Account</table>
        </columns>
        <columns>
            <checkedByDefault>false</checkedByDefault>
            <field>Owner.IsActive</field>
            <table>Account</table>
        </columns>
        <masterLabel>Accounts</masterLabel>
    </sections>
    <sections>
```

```
        <columns>
            <checkedByDefault>false</checkedByDefault>
            <field>Owner.Email</field>
            <table>Account.Contacts</table>
        </columns>
        <columns>
            <checkedByDefault>false</checkedByDefault>
            <field>byr__c</field>
            <table>Account.Contacts</table>
        </columns>
        <columns>
            <checkedByDefault>true</checkedByDefault>
            <field>ReportsTo.CreatedBy.Contact.Owner.MobilePhone</field>
            <table>Account.Contacts</table>
        </columns>
        <masterLabel>Contacts</masterLabel>
    </sections>
</ReportType>
```

Usage

The custom report type refers to fields by using their API names. For a historical field (one that has `trackTrending` set to `true`) the API name includes `hst`, such as `Field2__c_hst`.

```
<sections>
    <columns>
        <checkedByDefault>false</checkedByDefault>
        <field>Field2__c_hst</field>
        <table>CustomTrendedObject__c.CustomTrendedObject__c_hst</table>
    </columns>
    <masterLabel>History</masterLabel>
</sections>
```

For more information, see `trackTrending` on page 347.

Wildcard Support in the Manifest File

This metadata type supports the wildcard character `*` (asterisk) in the `package.xml` manifest file. For information about using the manifest file, see Deploying and Retrieving Metadata with the Zip File.

Role

Represents a role in your organization.

Declarative Metadata File Suffix and Directory Location

The file suffix for role components is `.role` and components are stored in the `roles` directory of the corresponding package directory.

Version

Role components are available in API version 24.0 and later.

Fields

This metadata type extends to subtype RoleOrTerritory on page 818.

Field Name	Field Type	Description
fullName	string	The unique identifier for API access. The fullName can contain only underscores and alphanumeric characters. It must be unique, begin with a letter, not include spaces, not end with an underscore, and not contain two consecutive underscores. This field is inherited from the Metadata component. Corresponds to **Role Name** in the user interface.
parentRole	string	The role above this role in the hierarchy.

Declarative Metadata Sample Definition

The following is the definition of a role.

```
<?xml version="1.0" encoding="UTF-8"?>
<Role xmlns="http://soap.sforce.com/2006/04/metadata">
    <caseAccessLevel>Edit</caseAccessLevel>
    <contactAccessLevel>Edit</contactAccessLevel>
    <description>Sample Role</description>
    <mayForecastManagerShare>false</mayForecastManagerShare>
    <name>R22</name>
    <opportunityAccessLevel>Read</opportunityAccessLevel>
</Role>
```

Wildcard Support in the Manifest File

This metadata type supports the wildcard character * (asterisk) in the package.xml manifest file. For information about using the manifest file, see Deploying and Retrieving Metadata with the Zip File.

RoleOrTerritory

Represents the common base type and valid values for role or territory.

Version

RoleOrTerritory components are available in API version 24.0 and later.

Note: You can’t create a RoleOrTerritory component directly. Use the Role or Territory metadata types instead.

Fields

Field Name	Field Type	Description
`caseAccessLevel`	string	Specifies whether a user can access other users' cases that are associated with accounts the user owns. Valid values are: • `Read` • `Edit` • `None` This field is not visible if your organization's sharing model for cases is Public Read/Write. If no value is set for this field, this field value uses the default access level that is specified in the Manage Territory page in Setup.
`contactAccessLevel`	string	Specifies whether a user can access other users' contacts that are associated with accounts the user owns. Valid values are: • `Read` • `Edit` • `None` This field is not visible if your organization's sharing model for contacts is Public Read/Write or Controlled by Parent. If no value is set for this field, this field value uses the default access level that is specified in the Manage Territory page in Setup.
`description`	string	The description of the role or territory.
`fullName`	string	The unique identifier for API access. The `fullName` can contain only underscores and alphanumeric characters. It must be unique, begin with a letter, not include spaces, not end with an underscore, and not contain two consecutive underscores. This field is inherited from the Metadata component.
`mayForecastManagerShare`	boolean	Indicates whether the forecast manager can manually share their own forecast.
`name`	string	Required. The name of the role or territory.
`opportunityAccessLevel`	string	Specifies whether a user can access other users' opportunities that are associated with accounts the user owns. Valid values are: • `Read` • `Edit` • `None` This field is not visible if your organization's sharing model for opportunities is Public Read/Write. If no value is set for this field, this field value uses the default access level that is specified in the Manage Territory page in Setup.

Declarative Metadata Sample Definition

The following is the definition of a role.

```
<?xml version="1.0" encoding="UTF-8"?>
<Role xmlns="http://soap.sforce.com/2006/04/metadata">
    <caseAccessLevel>Edit</caseAccessLevel>
    <contactAccessLevel>Edit</contactAccessLevel>
    <description>Sample Role</description>
    <mayForecastManagerShare>false</mayForecastManagerShare>
    <name>R22</name>
    <opportunityAccessLevel>Read</opportunityAccessLevel>
</Role>
```

The following is the definition of a territory.

```
<?xml version="1.0" encoding="UTF-8"?>
<Territory xmlns="http://soap.sforce.com/2006/04/metadata">
    <accountAccessLevel>Edit</accountAccessLevel>
    <caseAccessLevel>Edit</caseAccessLevel>
    <contactAccessLevel>Edit</contactAccessLevel>
    <description>Sample Territory</description>
    <mayForecastManagerShare>false</mayForecastManagerShare>
    <name>T22name</name>
    <opportunityAccessLevel>Read</opportunityAccessLevel>
</Territory>
```

SEE ALSO:

Role

Territory

SamlSsoConfig

Represents a SAML Single Sign-On configuration. This type extends the Metadata metadata type and inherits its `fullName` field. Single sign-on (SSO) lets users access authorized network resources with one login. You validate usernames and passwords against your corporate user database or other client app rather than Salesforce managing separate passwords for each resource.

File Suffix and Directory Location

SamlSsoConfig components have the suffix `.samlssoconfig` and are stored in the `samlssoconfigs` folder.

Version

SamlSsoConfig components are available in API version 28.0 and later.

Fields

Field Name	Field Type	Description
attributeNameIdFormat	string	For SAML 2.0, only and when identityLocation is set to Attribute. Possible values include unspecified, emailAddress or persistent. All legal values can be found in the "Name Identifier Format Identifiers" section of the Assertions and Protocols SAML 2.0 specification.
attributeName	string	The name of the identity provider's application. Get this name from your identity provider.
decryptionCertificate	string	The name of the certificate to use for decrypting incoming SAML assertions. This certificate is saved in the organization's Certificate and Key Management list. Available in API version 30.0 and later.
errorUrl	string	The URL of the page to direct users if there's an error during SAML login. It must be a publicly accessible page, such as a public site Visualforce page. The URL can be absolute or relative.
executionUserId	string	The user that runs the Apex handler class. The user must have the Manage Users permission. If you specify a SAML JIT handler class, a user is required.
identityLocation	SamlIdentityLocationType (enumeration of type string)	The location in the assertion where a user is identified. Valid values are: • SubjectNameId — The identity is in the <Subject> statement of the assertion. • Attribute — The identity is specified in an <AttributeValue>, located in the <Attribute> of the assertion.
identityMapping	SamlIdentityType (enumeration of type string)	The identifier the service provider uses for the user during Just-in-Time user provisioning. Valid values are: • Username — The user's Salesforce username. • FederationId — The federation ID from the user object; the identifier used by the service provider for the user. • UserId — The user ID from the user's Salesforce organization.
issuer	string	The identification string for the Identity Provider.
loginUrl	string	For SAML 2.0 only: The URL where Salesforce sends a SAML request to start the login sequence.
logoutUrl	string	For SAML 2.0 only: The URL to direct the user to when they click the Logout link. The default is http://www.salesforce.com.
name	string	The unique name used by the API and managed packages. The name must begin with a letter and use only alphanumeric characters and underscores. The name cannot end with an underscore or have two consecutive underscores.

Field Name	Field Type	Description
`oauthTokenEndpoint`	string	For SAML 2.0 only: The ACS URL used with enabling Salesforce as an identity provider in the web single sign-on OAuth assertion flow.
`redirectBinding`	boolean	If you're using My Domain, choose the binding mechanism your identity provider requests for your SAML messages. Values are: • `HTTP POST` — HTTP POST binding sends SAML messages using base64-encoded HTML forms. • `HTTP Redirect` — HTTP Redirect binding sends base64-encoded and URL-encoded SAML messages within URL parameters.
`requestSignatureMethod`	string	The method that's used to sign the SAML request. Valid values are `RSA-SHA1` and `RSA-SHA256`.
`requestSigningCertId`	string	The 18-digit ID for the certificate used to generate the signature on a SAML request to the identity provider. The certificate is saved in the Certificate and Key Management page in Setup.
`salesforceLoginUrl`	string	The URL associated with login for the web single sign-on flow. Note: When encryption is enabled, the URL has a parameter containing the ID of the SAML configuration, `sc=`***`samlSsoConfigId`***. For example, `https://mycompany.my.salesforce.com?sc=0LEB0000000CCC`. This change applies to API Version 47.0 and later.
`samlEntityId`	string	The issuer in SAML requests generated by Salesforce, and is also the expected audience of any inbound SAML Responses. If you don't have domains deployed, this value is always `https://saml.salesforce.com`. If you have domains deployed, Salesforce recommends that you use your custom domain name.
`samlJitHandlerId`	string	The name of an existing Apex class that implements the `Auth.SamlJitHandler` interface.
`samlVersion`	SamlType (enumeration of type string)	The SAML version in use. Valid values are: • `SAML1_1` — SAML 1.1 • `SAML2_0` — SAML 2.0
`singleLogoutBinding`	SamlSpSLOBinding (enumeration of type string)	The HTTP binding type. This value determines where to put the LogoutRequest or LogoutResponse in the SAML request during single logout (SLO). The value is base64 encoded. Valid values are: • `RedirectBinding` — Sent in the query string, deflated. • `PostBinding` — Sent in the POST body, not deflated.
`singleLogoutUrl`	string	The SAML single logout endpoint. This URL is the endpoint where Salesforce sends LogoutRequests (when Salesforce initiates a logout), or LogoutResponses (when the identity provider initiates a logout).

Field Name	Field Type	Description
useConfigRequestMethod	boolean	If true, applies the selected Request Signature Method (RSM) during single logout. If false, the default RSM (RSA-SHA1) is applied.
userProvisioning	boolean	If true, Just-in-Time user provisioning is enabled, which creates users on the fly the first time they try to log in. Specify Federation ID for the identityMapping value to use this feature.
validationCert	string	The certificate used to validate the request. Get this certificate from your identity provider.

Declarative Metadata Sample Definition

The following is an example of a SamlSsoConfig component. The validation certificate string has been truncated for readability.

```
<?xml version="1.0" encoding="UTF-8"?>
<SamlSsoConfig xmlns="http://soap.sforce.com/2006/04/metadata">
  <identityLocation>SubjectNameId</identityLocation>
  <identityMapping>FederationId</identityMapping>
  <issuer>https://my-idp.my.salesforce.com</issuer>
  <loginUrl>
    https://my-idp.my.salesforce.com/idp/endpoint/HttpRedirect
  </loginUrl>
  <logoutUrl>https://www.salesforce.com</logoutUrl>
  <name>SomeCompany</name>
  <oauthTokenEndpoint>
    https://login.salesforce.com/services/oauth2/token?so=00DD0000000
  </oauthTokenEndpoint>
  <redirectBinding>true</redirectBinding>
  <requestSignatureMethod>RSA-SHA1</requestSignatureMethod>
  <salesforceLoginUrl>
    https://login.salesforce.com?so=00DD0000000JxeI
  </salesforceLoginUrl>
  <samlEntityId>
    https://saml.salesforce.com/customPath
  </samlEntityId>
  <samlVersion>SAML2_0</samlVersion>
  <useConfigRequestMethod>true</useConfigRequestMethod>
  <userProvisioning>false</userProvisioning>
  <validationCert>
    MIIEojCCA4qgAwIBAgIOATtxsoBFAAAAAD4...
  </validationCert>
</SamlSsoConfig>
```

Wildcard Support in the Manifest File

This metadata type supports the wildcard character * (asterisk) in the package.xml manifest file. For information about using the manifest file, see Deploying and Retrieving Metadata with the Zip File.

Scontrol

Deprecated. Represents an Scontrol component, corresponding to an s-control in the Salesforce user interface.

Important: Visualforce pages supersede s-controls. Organizations that haven't previously used s-controls can't create them. Existing s-controls are unaffected, and can still be edited.

For more information, see "About S-Controls" in the Salesforce online help. This type extends the MetadataWithContent metadata type and inherits its `content` and `fullName` fields.

Declarative Metadata File Suffix and Directory Location

The file suffix is `.scf` for the s-control file. The accompanying metadata file is named *`ScontrolName`*`-meta.xml`.

Scontrol components are stored in the `scontrols` folder in the corresponding package directory.

Version

Scontrols are available in API version 10.0 and later.

Fields

This metadata type contains the following fields:

Field Name	Field Type	Description
`content`	base64Binary	Content of the s-control. Base 64-encoded binary data. Prior to making an API call, client applications must encode the binary attachment data as base64. Upon receiving a response, client applications must decode the base64 data to binary. This conversion is usually handled for you by a SOAP client. This field is inherited from the MetadataWithContent component.
`contentSource`	SControlContentSource (enumeration of type string)	Required. Determines how you plan to use the s-control: • `HTML`: Select this option if you want to enter the content for your s-control in `content`. • `URL`: Select this option if you want to enter the link or URL of an external website in `content`. • `Snippet`: Snippets are s-controls that are designed to be included in other s-controls. Select this option if you want to enter the content for your s-control snippet in `content`.
`description`	string	Optional text that describes the s-control. This only displays to users with "View All Data" permission (administrator).
`encodingKey`	Encoding (enumeration of type string)	Required. The default encoding setting is Unicode: `UTF-8`. Change it if you are passing information to a URL that requires data in a different format. This option is available when you select `URL` as the value for `contentSource`.

Field Name	Field Type	Description
fileContent	base64	File contents displayed if you add this s-control to a custom link. The file can contain a Java applet, Active-X control, or any other type of content you want. This option only applies to s-controls with a value of HTML for contentSource.
fileName	string	The unique name for the s-control. This name can contain only underscores and alphanumeric characters, and must be unique in your org. It must begin with a letter, not include spaces, not end with an underscore, and not contain two consecutive underscores. This field cannot be changed for components installed by a managed package. It is only relevant if the fileContent field also has a value. This is a new field in API version 14.0.
fullName	string	The s-control developer name used as a unique identifier for API access. The fullName can contain only underscores and alphanumeric characters. It must be unique, begin with a letter, not include spaces, not end with an underscore, and not contain two consecutive underscores. If this field contained characters before version 14.0 that are no longer allowed, the characters were stripped out of this field, and the previous value of the field was saved in the name field. This field is inherited from the Metadata component.
name	string	Required. The unique name for the s-control. It must contain alphanumeric characters only and begin with a letter. For example example_s_control.
supportsCaching	boolean	Required. Indicates whether the s-control supports caching (true) or not (false). Caching optimizes the page so that it remembers which s-controls are on the page when it reloads. This option only applies to HTML s-controls.

Declarative Metadata Sample Definition

The following sample creates the Myriad_Publishing.scf s-control, which creates a link to the website specified in the s-control. The corresponding Myriad_Publishing.scf-meta.xml metadata file follows the s-control file.

Myriad_Publishing.scf file:

```
http://www.myriadpubs.com
```

Myriad_Publishing.scf-meta.xml:

```
<?xml version="1.0" encoding="UTF-8"?>
<Scontrol xmlns="http://soap.sforce.com/2006/04/metadata">
    <contentSource>URL</contentSource>
    <description>s-control to open Myriad Publishing website.</description>
    <encodingKey>UTF-8</encodingKey>
    <name>Myriad Publishing</name>
    <supportsCaching>true</supportsCaching>
</Scontrol>
```

Wildcard Support in the Manifest File

This metadata type supports the wildcard character `*` (asterisk) in the `package.xml` manifest file. For information about using the manifest file, see Deploying and Retrieving Metadata with the Zip File.

ServiceChannel

Represents a channel of work items that are received from your organization—for example, cases, chats, or leads. This type extends the Metadata metadata type and inherits its `fullName` field.

File Suffix and Directory Location

ServiceChannel components have the suffix `.serviceChannel` and are stored in the `serviceChannels` folder.

Version

ServiceChannel components are available in API version 44.0 and later.

Special Access Rules

This type is available only if Omni-Channel is enabled in your org.

Fields

Field Name	Field Type	Description
`doesMinimizeWidgetOnAccept`	boolean	Automatically minimizes the Omni-Channel widget when an agent accepts work. This field is available in API version 48.0 and later.
`interactionComponent`	string	The custom console component to open in the footer when an agent accepts a work item from this service channel.
`label`	string	Required. The label of the service channel.
`relatedEntityType`	string	Required. The type of object that's associated with this service channel.
`secondaryRoutingPriorityField`	string	The name of the standard field or the ID of the custom field that is used for secondary routing priority. This field is available in API version 47.0 and later.
`serviceChannelFieldPriorities`	ServiceChannelFieldPriority[]	Required. A set of mappings between secondary routing priority field values and priorities. This field is available in API version 47.0 and later.

ServiceChannelFieldPriority

Represents a secondary routing priority field value mapping. Available in API version 47.0 and later.

Field Name	Field Type	Description
`priority`	int	Required. The priority number assigned to the mapped field value.
`value`	string	Required. The value of the secondaryRoutingPriorityField field defined in the parent ServiceChannel.

Declarative Metadata Sample Definition

The following is an example of a ServiceChannel component.

```
<?xml version="1.0" encoding="UTF-8"?>
<ServiceChannel xmlns="http://soap.sforce.com/2006/04/metadata">
    <doesMinimizeWidgetOnAccept>true</doesMinimizeWidgetOnAccept>
    <interactionComponent>ConsoleComponent</interactionComponent>
    <label>Case</label>
    <relatedEntityType>Case</relatedEntityType>
    <secondaryRoutingPriorityField>Status</secondaryRoutingPriorityField>
    <serviceChannelFieldPriorities>
        <priority>1</priority>
        <value>Escalated</value>
    </serviceChannelFieldPriorities>
    <serviceChannelFieldPriorities>
        <priority>2</priority>
        <value>On Hold</value>
    </serviceChannelFieldPriorities>
</ServiceChannel>
```

The following is an example `package.xml` that references the previous definition.

```
<?xml version="1.0" encoding="UTF-8"?>
<Package xmlns="http://soap.sforce.com/2006/04/metadata">
    <types>
        <members>*</members>
        <name>ServiceChannel</name>
    </types>
    <version>44.0</version>
</Package>
```

Wildcard Support in the Manifest File

This metadata type supports the wildcard character `*` (asterisk) in the `package.xml` manifest file. For information about using the manifest file, see Deploying and Retrieving Metadata with the Zip File.

ServicePresenceStatus

Represents a presence status that can be assigned to a service channel. This type extends the Metadata metadata type and inherits its `fullName` field.

File Suffix and Directory Location

ServicePresenceStatus components have the suffix `.servicePresenceStatus` and are stored in the `servicePresenceStatuses` folder.

Version

ServicePresenceStatus components are available in API version 44.0 and later.

Special Access Rules

This type is available only if Omni-Channel is enabled in your org.

Fields

Field Name	Field Type	Description
`channels`	ServiceChannelStatus	Represents the status that's associated with a specific service channel.
`label`	string	The label of the presence status.

ServiceChannelStatus

Represents the status that's associated with a specific service channel.

Field Name	Field Type	Description
`channel`	string	Represents the channels assigned to the presence status.

Declarative Metadata Sample Definition

The following is an example of a ServicePresenceStatus component.

```
<?xml version="1.0" encoding="UTF-8"?>
<ServicePresenceStatus xmlns="http://soap.sforce.com/2006/04/metadata">
    <channels>
        <channel>Case</channel>
    </channels>
    <label>Available for Cases</label>
</ServicePresenceStatus>
```

The following is an example `package.xml` that references the previous definition.

```
<?xml version="1.0" encoding="UTF-8"?>
<Package xmlns="http://soap.sforce.com/2006/04/metadata">
    <types>
        <members>*</members>
        <name>ServicePresenceStatus</name>
    </types>
    <version>44.0</version>
</Package>
```

Wildcard Support in the Manifest File

This metadata type supports the wildcard character `*` (asterisk) in the `package.xml` manifest file. For information about using the manifest file, see Deploying and Retrieving Metadata with the Zip File.

Settings

Represents the organization settings related to a feature. For example, your password policies, session settings and network access controls are all available in the SecuritySettings component type.

Not all feature settings are available in the Metadata API. See Unsupported Metadata Types on page 136 for information on which feature settings are not available.

Settings can be accessed using the specific component member or via wildcard. For example, in the package manifest file you would use the following section to access SecuritySettings:

```
    <types>
        <members>Security</members>
        <name>Settings</name>
    </types>
```

The member format when used in the package manifest is the component metadata type name without the “Settings” suffix, so in the preceding example “Security” is used instead of “SecuritySettings”.

File Suffix and Directory Location

Each settings component gets stored in a single file in the `settings` directory of the corresponding package directory. The filename uses the format *`Setting feature`*`.settings`. For example, the SecuritySettings file would be `Security.settings`. See “File Suffix and Directory Location” information for the individual settings components to determine the exact filename.

Version

Settings is available in API version 27.0 and later. See the version information for the individual setting component to determine which API version the settings component became available.

Declarative Metadata Sample Definition

The following is an example package manifest used to deploy or retrieve only the MobileSettings for an organization:

```
<?xml version="1.0" encoding="UTF-8"?>
<Package xmlns="http://soap.sforce.com/2006/04/metadata">
    <types>
        <members>Mobile</members>
        <name>Settings</name>
    </types>
    <version>27.0</version>
</Package>
```

The following is an example package manifest used to deploy or retrieve all the available settings metadata for an organization, using a wildcard:

```
<?xml version="1.0" encoding="UTF-8"?>
<Package xmlns="http://soap.sforce.com/2006/04/metadata">
    <types>
        <members>*</members>
        <name>Settings</name>
    </types>
    <version>27.0</version>
</Package>
```

IN THIS SECTION:

AccountSettings

Represents an org's account settings for account teams, account owner report, and the **View Hierarchy** link. This type extends the Metadata metadata type and inherits its `fullName` field.

AccountInsightsSettings

Represents an org's Einstein Account Insights settings. This setting controls features that help your reps maintain their relationships with their customers. This type extends the Metadata metadata type and inherits its `fullName` field.

AccountIntelligenceSettings

Represents an org's Account Intelligence settings. These settings control features that make it easy for sales reps to create accounts, see relevant news articles, and add logos to account records. This type extends the Metadata metadata type and inherits its `fullName` field.

ActionsSettings

Represents an org's actions settings for default quick actions, multi-dimensional publisher, and third-party actions. This type extends the Metadata metadata type and inherits its `fullName` field.

ActivitiesSettings

Represents an org's activity settings, and its user interface settings for the calendar. This type extends the Metadata metadata type and inherits its `fullName` field.

AddressSettings

Represents the configuration of country and state picklists. Use the AddressSettings component type to configure state and country data in your organization so that you can convert text-based values into standard picklist values. To convert your state and country values, from Setup, enter *State and Country Picklists* in the `Quick Find` box, then select **State and Country Picklists**. For more information, see "Let Users Select State and Country from Picklists" in the Salesforce online help.

AnalyticsSettings

Represents Analytics settings in your Salesforce org. Einstein Analytics allows you to explore all of your data quickly and easily by providing AI-powered advanced Analytics, right inside Salesforce. Manage your datasets, query data with Salesforce Analytics Query Language (SAQL), and customize dashboards. You can use these settings to configure which Analytics features are available to users in your org. This type extends the Metadata metadata type and inherits its `fullName` field.

ApexSettings

Represents Apex-related org settings. This type extends the Metadata metadata type and inherits its `fullName` field.

AppExperienceSettings

Represents settings for the app experience.This type extends the Metadata metadata type and inherits its `fullName` field.

AutomatedContactsSettings

Represents an org's Einstein Automated Contacts settings. These settings let you find new contacts and opportunity contact roles. This type extends the Metadata metadata type and inherits its `fullName` field.

BotSettings

Represents an organization's Einstein Bot settings, such as whether or not Einstein Bots is enabled. This type extends the Metadata metadata type and inherits its `fullName` field.

BusinessHoursSettings

Represents the metadata used to manage settings for business hours and holidays in entitlements, entitlement templates, campaigns, and cases. This type extends the Metadata metadata type and inherits its `fullName` field.

CampaignSettings

Represents an org's Campaign Influence settings, which help you understand how your campaigns are affecting your opportunity pipeline. This type extends the Metadata metadata type and inherits its `fullName` field.

CaseClassificationSettings

Represents the Einstein Case Classification settings for an org. This type extends the Metadata metadata type and inherits its `fullName` field.

CaseSettings

Represents an organization's case settings, such as the default case owner, which case-related features are enabled, and which Classic email templates are used for various case activities. This type extends the Metadata metadata type and inherits its `fullName` field.

ChatterAnswersSettings

Represents the metadata used to manage settings for Chatter Answers.

ChatterEmailMDSettings

Represents an org's settings for Chatter email when Chatter is enabled. This type extends the Metadata metadata type and inherits its `fullName` field.

ChatterSettings

Represents an org's settings for their Chatter instance when Chatter is enabled for the org. This type extends the Metadata metadata type and inherits its `fullName` field.

CommunitiesSettings

Represents the feed moderation settings for an org. This type extends the Metadata metadata type and inherits its `fullName` field.

CompanySettings

Represents global settings that affect multiple features in your organization. This type extends the Metadata metadata type and inherits its `fullName` field.

ConnectedAppSettings

Represents settings for connected apps. This type extends the Metadata metadata type and inherits its `fullName` field.

ContentSettings

Represents content settings for an org. This type extends the Metadata metadata type and inherits its `fullName` field.

ContractSettings

Represents contract settings. For more information, see “Set Up Contracts” in the Salesforce online help.

CurrencySettings

Represents an organization's currency settings, including supporting multiple currencies and currency effective dates. This type extends the Metadata metadata type and inherits its `fullName` field.

DataDotComSettings

Represents the org's Data.com settings. This type extends the Metadata metadata type and inherits its `fullName` field.

DeploymentSettings

Represents the settings affecting how deployments behave in the org. This type extends the Metadata metadata type and inherits its `fullName` field.

DevHubSettings

Represents Developer Hub (Dev Hub) settings. This type extends the Metadata metadata type and inherits its `fullName` field.

EACSettings

Represents the Einstein Activity Capture metadata type. Use Einstein Activity Capture to add emails and events from your Microsoft or Google account to the activity timeline of related Salesforce records. Automatically sync contact and event data between your Microsoft or Google account and Salesforce. This type extends the Metadata metadata type and inherits its `fullName` field.

EmailAdministrationSettings

Represents an organization's email administration settings, including email deliverability, security compliance, relay configurations, and system notifications. This type extends the Metadata metadata type and inherits its `fullName` field.

EmailIntegrationSettings

Represents an org's settings for the Outlook integration, Gmail integration, and Salesforce Inbox. This type extends the Metadata metadata type and inherits its `fullName` field.

EmailTemplateSettings

Represents an org's email template settings. This type extends the Metadata metadata type and inherits its `fullName` field.

EnhancedNotesSettings

Represents an org's enhanced note settings, such as enabling enhanced notes and enabling tasks in enhanced notes.This type extends the Metadata metadata type and inherits its `fullName` field.

EncryptionKeySettings

Represents an org's encryption key settings, such as customer-supplied keys options and key derivation settings. This type extends the Metadata metadata type and inherits its `fullName` field.

EntitlementSettings

Represents an organization's entitlement settings.

EventSettings

Represents an org's platform event settings.

ExperienceBundleSettings

Represents the settings of ExperienceBundleMetadata of an organization. This type extends the Metadata metadata type and inherits its `fullName` field.

FieldServiceSettings

Represents an organization's Field Service Lightning settings.

FileUploadAndDownloadSecuritySettings

Represents the security settings for uploading and downloading files. This type extends the Metadata metadata type and inherits its `fullName` field.

FlowSettings

Represents the org's settings for processes and flows, such as whether Lightning runtime for flows is enabled. This type extends the Metadata metadata type and inherits its `fullName` field.

ForecastingSettings

Represents the Collaborative Forecasts settings options. This type extends the Metadata metadata type and inherits its `fullName` field.

HighVelocitySalesSettings

Represents an org's High Velocity Sales settings. With High Velocity Sales, you can make your inside sales team as effective as possible. This type extends the Metadata metadata type and inherits its `fullName` field.

IdeasSettings

Represents the metadata used to manage settings for Ideas.

IndustriesSettings

Represents settings for industries verticals like Financial Service Cloud, Consumer Goods Cloud, and Health Cloud.This type extends the Metadata metadata type and inherits its `fullName` field.

InvocableActionSettings

Represents the org's invocable action settings, such as whether partial save is allowed.This type extends the Metadata metadata type and inherits its `fullName` field.

IoTSettings

Represents the organization's IoT settings, such as whether or not IoT or IoT Insights is enabled.This type extends the Metadata metadata type and inherits its `fullName` field.

KnowledgeSettings

Represents the metadata used to manage settings for Salesforce Knowledge.

LanguageSettings

Represents an organization's language settings. Language settings control end-user language selection, locale formats, and translation options. This type extends the Metadata metadata type and inherits its `fullName` field.

LeadConfigSettings

Represents configuration settings for Leads that control how they are converted and displayed, and what actions are available. This type extends the Metadata metadata type and inherits its `fullName` field.

LeadConvertSettings

Represents an organization's custom field mappings for lead conversion. Custom fields can be mapped from Leads to Accounts, Contacts, and Opportunities. Options for creating opportunities during lead conversion can also be specified. This type extends the Metadata metadata type and inherits its `fullName` field.

LiveAgentSettings

Represents an organization's Chat settings, such as whether or not Chat is enabled. This type extends the Metadata metadata type and inherits its `fullName` field.

LightningExperienceSettings

Represents the settings that modify an org's Lightning Experience configuration. This type extends the Metadata metadata type and inherits its `fullName` field.

LiveMessageSettings

Represents an org's LiveMessage settings.

MacroSettings

Represents an organization's Macro settings, such as whether or not folders is enabled. This type extends the Metadata metadata type and inherits its `fullName` field.

MapAndLocationSettings

Represents an org's map and location settings.

MobileSettings

Represents an organization's mobile settings. This type extends the Metadata metadata type and inherits its `fullName` field.

MyDomainSettings

Represents your org's My Domain settings. My Domain is a Salesforce Identity feature that lets you create your own subdomain to better manage login and authentication. With a My Domain subdomain, you replace the instance name that Salesforce assigned you, such as `https://na00/salesforce.com` with a name that you choose, such as `https://`*`yourcompany`*`.my.salesforce.com`. With My Domain, you can create a custom login page and custom login policies to control how users are authenticated. You can also set up single sign-on (SSO) and social sign-on, work in multiple orgs at the same time, and more. This type extends the Metadata metadata type and inherits its `fullName` field.

NameSettings

Enables or disables the formal name, middle name, and suffix attributes for the following person objects: Contact, Lead, Person Account, and User.This type extends the Metadata metadata type and inherits its `fullName` field.

NotificationsSettings

Represents an organization's mobile settings.

ObjectLinkingSettings (Beta)

Represents the channel-object linking settings for an org. This type extends the Metadata metadata type and inherits its `fullName` field.

OmniChannelSettings

Represents the Omni-Channel settings for an org. This type extends the Metadata metadata type and inherits its `fullName` field.

OpportunityInsightsSettings

Represents an org's Einstein Opportunity Insights settings. This setting controls features that give you relevant updates about your opportunities. This type extends the Metadata metadata type and inherits its `fullName` field.

OpportunitySettings

Represents org preferences for features such as automatic opportunity updates and similar-opportunity filters. This type extends the Metadata metadata type and inherits its `fullName` field.

OrderManagementSettings

Represents options for the Salesforce Order Management product.This type extends the Metadata metadata type and inherits its `fullName` field.

OrderSettings

Represents order settings. This type extends the Metadata metadata type and inherits its `fullName` field.

OrgPreferenceSettings

Removed in API version 48.0. Represents the unique org preference settings in a Salesforce org.

OrgSettings

Represents the settings for org-wide functionality that isn't associated with any specific feature.This type extends the Metadata metadata type and inherits its `fullName` field.

PartyDataModelSettings

Represents an organization's party data model settings, including options around the Individual object and consent enablement. This type extends the Metadata metadata type and inherits its `fullName` field.

PardotSettings

Represents Pardot settings in your Salesforce org. Pardot is a powerful B2B marketing automation solution that helps you create meaningful connections, generate more pipeline, and empower sales to close more deals. You can use these settings to configure how Pardot collects and displays data.This type extends the Metadata metadata type and inherits its `fullName` field.

PardotEinsteinSettings

Represents PardotEinsteinSettings. Use these settings to learn what factors drive your campaign performance, and get the best possible engagement score for your prospects. This type extends the Metadata metadata type and inherits its `fullName` field.

PathAssistantSettings

Represents the Path preference setting. This type extends the Metadata metadata type and inherits its `fullName` field.

PicklistSettings

Represents an org's picklist settings. These settings control the behavior of a picklist. This type extends the Metadata metadata type and inherits its `fullName` field.

PlatformEncryptionSettings

Represents an org's Platform Encryption settings, such as settings for available encryption schemes, permissions, encryption policy access, and which fields can be encrypted. This type extends the Metadata metadata type and inherits its `fullName` field.

PredictionBuilderSettings

Represents the settings that determine how a user can interact with Einstein Prediction Builder. This type extends the Metadata metadata type and inherits its `fullName` field.

PrivacySettings

Represents an organization's settings for data privacy and consent management. This type extends the Metadata metadata type and inherits its `fullName` field.

ProductSettings

Represents organization preferences for quantity schedules, revenue schedules, and active flag interaction with prices. This type extends the Metadata metadata type and inherits its `fullName` field.

QuoteSettings

Represents an org's quotes settings, such as enabling quotes or creating quotes without an associated opportunity. This type extends the Metadata metadata type and inherits its `fullName` field.

RecordPageSettings

Represents an org's record page settings. This type extends the Metadata metadata type and inherits its `fullName` field.

SchemaSettings

Represents an org's schema settings, which manage the availability of custom settings and custom metadata type values. This type extends the Metadata metadata type and inherits its `fullName` field.

SearchSettings

Represents an org's search settings. This type extends the Metadata metadata type and inherits its `fullName` field.

SecuritySettings

Represents an org's security settings. For example, settings define trusted IP ranges for network access, password and login requirements, session expiration, and single sign-on settings. This type extends the Metadata metadata type and inherits its `fullName` field.

SharingSettings

Represents an organization's sharing, visibility, and data access settings. This type extends the Metadata metadata type and inherits its `fullName` field.

SiteSettings

Represents the settings for Salesforce Sites. This type extends the Metadata metadata type and inherits its `fullName` field.

SocialCustomerServiceSettings

Represents Social Customer Service settings such as how to format inbound content from social posts to cases.This type extends the Metadata metadata type and inherits its `fullName` field.

SocialProfileSettings

Represents org preferences for social media features such as enabling Twitter and Facebook. This type extends the Metadata metadata type and inherits its `fullName` field.

SurveySettings

Represents an org's survey settings. Use the SurveySettings component to enable surveys for your org and choose whether the owner of a survey can manage the responses. This type extends the Metadata metadata type and inherits its `fullName` field.

Territory2Settings

Represents an org's Territory2 settings. Use Territory2 settings to set the access level that Territory Management 2.0 users have to records associated with sales territories, and to enable features. The standard record access settings apply to accounts and opportunities. If your Salesforce org uses *`Private`* default internal access for contacts or cases, you can also set access for those records. This type extends the Metadata metadata type and inherits its `fullName` field.

TrailheadSettings

Represents an org's access to myTrailhead. This type extends the Metadata metadata type and inherits its `fullName` field.

TrialOrgSettings

Represents the settings in a trial user's org. This type extends the Metadata metadata type and inherits its `fullName` field.

UserEngagementSettings

Represents the metadata associated with various feature settings around Lightning Experience transition and adoption, user engagement and adoption assistance, and adoption apps. This type extends the Metadata metadata type and inherits its `fullName` field.

UserInterfaceSettings

Represents the settings that modify the behavior of the org's user interface. This type extends the Metadata metadata type and inherits its `fullName` field.

UserManagementSettings

Represents a selection of user management options that appear on the User Management Settings Setup page. This type extends the Metadata metadata type and inherits its `fullName` field.

VoiceSettings

Represents an org's Lightning Dialer settings, such as call recording, conferencing, and voicemail. This type extends the Metadata metadata type and inherits its `fullName` field.

WorkDotComSettings

Represents WorkDotCom settings. This type extends the Metadata metadata type and inherits its `fullName` field.

AccountSettings

Represents an org's account settings for account teams, account owner report, and the **View Hierarchy** link. This type extends the Metadata metadata type and inherits its `fullName` field.

In the package manifest, all organization settings metadata types are accessed using the "Settings" name. See Settings for more details.

File Suffix and Directory Location

AccountSettings values are stored in the `Account.settings` file in the `settings` folder. The `.settings` files are different from other named components because there is only one settings file for each settings component.

Version

AccountSettings is available in API versions 29.0 and later.

Fields

Field Name	Field Type	Description
`enableAccountHistoryTracking`	boolean	Indicates whether history tracking is enabled for accounts (`true`) or not (`false`). The default value is `false`. If history tracking is disabled, the History related list is removed from account page layouts. However, history data is still available for reporting up to the date and time when tracking was disabled. Available in API version 47.0 and later.
`enableAccountOwnerReport`	boolean	Indicates whether Account Owner Report can (`true`) or can't (`false`) be run by all users.
`enableAccountInsightsInMobile`	boolean	Indicates whether users can see Einstein Account Insights on their mobile device (`true`) or not (`false`). Insights appear in the Einstein Insights component, which is on account records and the Home page. Available in API version 47.0 and later. To use this feature, users must have the Einstein Account Insights permission.
`enableAccountTeams`	boolean	Indicates whether account teams are enabled (`true`) or not (`false`). The Metadata API can't be used to disable account teams.
`enableContactHistoryTracking`	boolean	Indicates whether history tracking is enabled for contacts (`true`) or not (`false`). Available in API version 46.0 and later.
`enableRelateContactToMultipleAccounts`	boolean	Indicates whether users can relate a contact to multiple accounts (`true`) or only one account (`false`). The default value is `false`. If this feature (Contacts to Multiple Accounts) is disabled, secondary contact–account relationships created while the feature was enabled are deleted. Available in API version 47.0 and later.
`showViewHierarchyLink`	boolean	Indicates whether the default **View Hierarchy** link on all business account detail pages is visible (`true`) or hidden (`false`).

Declarative Metadata Sample Definition

The following is an example of the Account.settings file:

```xml
<?xml version="1.0" encoding="UTF-8"?>
<AccountSettings xmlns="http://soap.sforce.com/2006/04/metadata">
 <enableAccountOwnerReport>true</enableAccountOwnerReport>
 <enableAccountTeams>true</enableAccountTeams>
 <enableAccountInsightsInMobile>true</enableAccountInsightsInMobile>
 <enableContactHistoryTracking>true</enableContactHistoryTracking>
 <showViewHierarchyLink>true</showViewHierarchyLink>
</AccountSettings>
```

Example Package Manifest

The following is an example package manifest used to deploy or retrieve the Account settings metadata:

```
<?xml version="1.0" encoding="UTF-8"?>
<Package xmlns="http://soap.sforce.com/2006/04/metadata">
 <types>
  <members>Account</members>
  <name>Settings</name>
 </types>
 <version>29.0</version>
</Package>
```

Wildcard Support in the Manifest File

The wildcard character `*` (asterisk) in the `package.xml` manifest file doesn't apply to metadata types for feature settings. The wildcard applies only when retrieving all settings, not for an individual setting. For details, see Settings. For information about using the manifest file, see Deploying and Retrieving Metadata with the Zip File.

AccountInsightsSettings

Represents an org's Einstein Account Insights settings. This setting controls features that help your reps maintain their relationships with their customers. This type extends the Metadata metadata type and inherits its `fullName` field.

In the package manifest, all organization settings metadata types are accessed using the "Settings" name. See Settings for more details.

File Suffix and Directory Location

AccountInsightsSettings values are stored in the `AccountInsights.settings` file in the `settings` folder. The `.settings` files are different from other named components because there is only one settings file for each settings component.

Version

AccountInsightsSettings is available in API versions 48.0 and later.

Fields

Field Name	Field Type	Description
`enableAccountInsights`	boolean	Indicates whether Einstein Account Insights is enabled (`true`) or not (`false`). The default value is `false`.

Declarative Metadata Sample Definition

The following is an example of the AccountInsights.settings file:

```
<?xml version="1.0" encoding="UTF-8"?>
<AccountInsightsSettings xmlns="http://soap.sforce.com/2006/04/metadata">
 <enableAccountInsights>true</enableAccountInsights>
</AccountInsightsSettings>
```

Example Package Manifest

The following is an example package manifest used to deploy or retrieve the AccountInsights settings metadata:

```
<?xml version="1.0" encoding="UTF-8"?>
<Package xmlns="http://soap.sforce.com/2006/04/metadata">
 <types>
  <members>AccountInsights</members>
  <name>Settings</name>
 </types>
 <version>29.0</version>
</Package>
```

Wildcard Support in the Manifest File

The wildcard character `*` (asterisk) in the `package.xml` manifest file doesn't apply to metadata types for feature settings. The wildcard applies only when retrieving all settings, not for an individual setting. For details, see Settings. For information about using the manifest file, see Deploying and Retrieving Metadata with the Zip File.

AccountIntelligenceSettings

Represents an org's Account Intelligence settings. These settings control features that make it easy for sales reps to create accounts, see relevant news articles, and add logos to account records. This type extends the Metadata metadata type and inherits its `fullName` field.

In the package manifest, all organization settings metadata types are accessed using the "Settings" name. See Settings for more details.

File Suffix and Directory Location

AccountIntelligenceSettings values are stored in the `AccountIntelligence.settings` file in the `settings` folder. The `.settings` files are different from other named components because there is only one settings file for each settings component.

Version

AccountIntelligenceSettings is available in API versions 48.0 and later.

Fields

Field Name	Field Type	Description
enableAccountLogos	boolean	Indicates whether your sales reps can see available company logos (true) or not (false). The logos are for US-based companies only. The default value is false. enableAutomatedAccountFields must be true to use this setting.
enableAutomatedAccountFields	boolean	Indicates whether Automated Account Fields is enabled (true) or not (false). The default value is false.
enableNewsStories	boolean	Indicates whether News is enabled (true) or not (false). The default value is false. enableAutomatedAccountFields must be true to use this setting.

Declarative Metadata Sample Definition

The following is an example of the AccountIntelligence.settings file:

```
<?xml version="1.0" encoding="UTF-8"?>
<AccountIntelligenceSettings xmlns="http://soap.sforce.com/2006/04/metadata">
 <enableAccountLogos>true</enableAccountLogos>
 <enableAutomatedAccountFields>true</enableAutomatedAccountFields>
 <enableNewsStories>true</enableNewsStories>
</AccountIntelligenceSettings>
```

Example Package Manifest

The following is an example package manifest used to deploy or retrieve the AccountIntelligence settings metadata:

```
<?xml version="1.0" encoding="UTF-8"?>
<Package xmlns="http://soap.sforce.com/2006/04/metadata">
 <types>
  <members>AccountIntelligence</members>
  <name>Settings</name>
 </types>
 <version>29.0</version>
</Package>
```

Wildcard Support in the Manifest File

The wildcard character * (asterisk) in the package.xml manifest file doesn't apply to metadata types for feature settings. The wildcard applies only when retrieving all settings, not for an individual setting. For details, see Settings. For information about using the manifest file, see Deploying and Retrieving Metadata with the Zip File.

ActionsSettings

Represents an org's actions settings for default quick actions, multi-dimensional publisher, and third-party actions. This type extends the Metadata metadata type and inherits its `fullName` field.

In the package manifest, all organization settings metadata types are accessed using the "Settings" name. See Settings for more details.

File Suffix and Directory Location

ActionsSettings values are stored in the `Actions.settings` file in the `settings` folder. The `.settings` files are different from other named components because there is only one settings file for each settings component.

Version

components are available in API version 47.0 and later.

Fields

Field Name	Field Type	Description
`enableDefaultQuickActionsOn`	boolean	Indicates whether default quick actions are created in the org (`true`, the default setting) or not (`false`).
`enableMdpEnabled`	boolean	Indicates whether multi-dimensional publisher is enabled (`true`, the default setting) or not (`false`).
`enableThirdPartyActions`	boolean	Indicates whether third-party actions are displayed in the multi-dimensional publisher (`true`) or not (`false`, the default setting).

Declarative Metadata Sample Definition

The following is an example of an ActionsSettings component.

```
<?xml version="1.0" encoding="UTF-8"?>
<ActionsSettings xmlns="http://soap.sforce.com/2006/04/metadata">
 <DefaultQuickActionsOn>true</DefaultQuickActionsOn>
 <MdpEnabled>true</MdpEnabled>
 <ThirdPartyActions>true</ThirdPartyActions>
</ActionsSettings>
```

Wildcard Support in the Manifest File

The wildcard character `*` (asterisk) in the `package.xml` manifest file doesn't apply to metadata types for feature settings. The wildcard applies only when retrieving all settings, not for an individual setting. For details, see Settings. For information about using the manifest file, see Deploying and Retrieving Metadata with the Zip File.

ActivitiesSettings

Represents an org's activity settings, and its user interface settings for the calendar. This type extends the Metadata metadata type and inherits its `fullName` field.

Use the ActivitiesSettings component type to control the following activity settings:

- Configure group and recurring tasks, recurring and multiday events, and email tracking
- Relate multiple contacts to tasks and events (shared activities)
- Display custom logos in meeting requests

Also use the ActivitiesSettings component type to control user interface settings for the calendar, including hover links and drag-and-drop editing.

In the package manifest, all organization settings metadata types are accessed using the “Settings” name. See Settings for more details.

File Suffix and Directory Location

ActivitiesSettings values are stored in the `Activities.settings` file in the `settings` directory. The `.settings` files are different from other named components because there is only one settings file for each settings component.

Version

ActivitiesSettings is available in API versions 28.0 and later.

Fields

Settings for all types listed below are controlled on the Activity settings page or the User Interface settings page as noted.

Field Name	Field Type	Description
`allowUsersToRelateMultipleContactsToTasksAndEvents`	boolean	This field indicates whether Shared Activities is enabled. When the value is true, allows users to relate multiple contacts to a task or event. Important: Beginning with API v36.0, this field is read-only in all versions of the API. You can’t change the value of this field. Even though this field was updateable before Spring '16, changing this field’s value wasn't supported and could have resulted in an incorrect integration. If you have code in older API versions that changes the value of this field, ensure that you update that code to prevent any errors.
`autoRelateEventAttendees`	boolean	When users add attendees to events, events are automatically related to up to 50 contacts or one lead. An attendee is matched by their email address to a contact or lead. Admins control this field on the Activity Settings page. Available in API version 42.0 and later.
`enableActivityReminders`	boolean	Enables popup activity reminders for an organization. Admins control this field on the Activity Settings page.
`enableClickCreateEvents`	boolean	Lets users create events in day and weekly calendar views by double-clicking a specific time slot and entering the details of the event in an overlay. Hovering over an event displays an overlay where users can view the event details or delete the event without leaving the page.

Field Name	Field Type	Description
		Admins use a mini page layout to configure the fields shown in the overlays. Does not support recurring events or multi-person events. Admins control this field on the User Interface settings page.
`enableDragAndDropScheduling`	boolean	Lets users create events associated with records by dragging a record from a list view onto a calendar view and entering the details of the event in an overlay. Hovering over an event displays an overlay where users can view the event details or delete the event without leaving the page. Admins use a mini page layout to configure the fields shown in the overlays. Admins control this field on the User Interface settings page.
`enableEmailTracking`	boolean	Enables tracking of outbound HTML emails if an organization uses HTML email templates. Admins control this field on the Activity settings page.
`enableGroupTasks`	boolean	Lets users assign independent copies of a new task to multiple users. Admins control this field on the Activity settings page.
`enableListViewScheduling`	boolean	Extends the functionality of `enableDragAndDropScheduling` and `enableClickCreateEvents` to list view calendars. Admins control this field on the User Interface settings page.
`enableMultidayEvents`	boolean	Enables creation of events that end more than 24 hours after they start. Admins control this field on the Activity settings page.
`enableRecurringEvents`	boolean	Enables creation of events that repeat at specified intervals. Admins control this field on the Activity settings page.
`enableRecurringTasks`	boolean	Enables creation of tasks that repeat at specified intervals. Admins control this field on the Activity settings page.
`enableRollUpActivToContactsAcct`	boolean	Enables a contact's activities to be rolled up and displayed on the contact's primary account. Default value is `true`. Available in API versions 47.0 and later.
`enableSidebarCalendarShortcut`	boolean	In the sidebar, displays a shortcut link to a user's last-used calendar view. Admins control this field on the Activity settings page.
`enableSimpleTaskCreateUI`	boolean	Allows Admins to specify whether tapping New Task in Salesforce opens a regular task record edit page or a page that displays key task fields first. Admins control this field on the Activity settings page.

Field Name	Field Type	Description
enableUNSTaskDelegatedToNotifications	boolean	On the Activity settings page, exposes a setting for Admins to hide or show a user setting that lets individual users enable or disable email notifications when tasks are assigned to them.
enableUserListViewCalendars	boolean	Allows users to create and view user list view calendars in Lightning Experience. Available in API versions 47.0 and later
meetingRequestsLogo	string	Available when showCustomLogoMeetingRequests is enabled. Uploads a custom logo. An administrator can select only a logo that has been uploaded to certain folders in the Documents tab. Admins control this field on the Activity settings page.
showCustomLogoMeetingRequests	boolean	Displays a custom logo in meeting request emails and on a meeting's Web page. Invitees see the logo when a user either invites them to an event or requests a meeting. Admins control this field on the Activity settings page.
showEventDetailsMultiUserCalendar	boolean	Displays event details on-screen rather than in hover text. Admins control this field on the Activity settings page.
showHomePageHoverLinksForEvents	boolean	In the calendar section of the Home tab: • When a user hovers over the subject of an event, a hover link displays an overlay with selected event details. (Hover links are always available in other calendar views.) • When a user clicks the subject of an event, displays the event detail page. Admins use a mini page layout to configure the fields shown in the overlay. Admins control this field on the User Interface settings page.
showMyTasksHoverLinks	boolean	In the My Tasks section of the Home tab and on the calendar day view: • When a user hovers over the subject of a task, a hover link displays an overlay with selected task details. • When a user clicks the subject of a task, displays the task detail page. Admins use a mini page layout to configure the fields shown in the overlay. Admins control this field on the User Interface settings page.

Example Package Manifest

The following is an example package manifest used to deploy or retrieve the Activity settings metadata for an organization:

```
<?xml version="1.0" encoding="UTF-8"?>
<Package xmlns="http://soap.sforce.com/2006/04/metadata">
```

```
    <types>
        <members>Activities</members>
        <name>Settings</name>
    </types>
    <version>28.0</version>
</Package>
```

Declarative Metadata Sample Definition

The following is an example of an activity settings file:

```
<?xml version="1.0" encoding="UTF-8"?>
<ActivitiesSettings xmlns="http://soap.sforce.com/2006/04/metadata">
    <enableActivityReminders>true</enableActivityReminders>
    <autoRelateEventAttendees>true</autoRelateEventAttendees>
    <enableClickCreateEvents>true</enableClickCreateEvents>
    <enableDragAndDropScheduling>true</enableDragAndDropScheduling>
    <enableEmailTracking>true</enableEmailTracking>
    <enableGroupTasks>true</enableGroupTasks>
    <enableListViewScheduling>true</enableListViewScheduling>
    <enableMultidayEvents>true</enableMultidayEvents>
    <enableRecurringEvents>true</enableRecurringEvents>
    <enableRollUpActivToContactsAcct>true</enableRollUpActivToContactsAcct>
    <enableRecurringTasks>true</enableRecurringTasks>
    <enableUserListViewCalendars>true</enableUserListViewCalendars>
    <enableSidebarCalendarShortcut>true</enableSidebarCalendarShortcut>
    <meetingRequestsLogo>Folder02/logo03.png</meetingRequestsLogo>
    <showCustomLogoMeetingRequests>true</showCustomLogoMeetingRequests>
    <showEventDetailsMultiUserCalendar>true</showEventDetailsMultiUserCalendar>
    <showHomePageHoverLinksForEvents>true</showHomePageHoverLinksForEvents>
    <showMyTasksHoverLinks>true</showMyTasksHoverLinks>
</ActivitiesSettings>
```

Wildcard Support in the Manifest File

The wildcard character `*` (asterisk) in the `package.xml` manifest file doesn't apply to metadata types for feature settings. The wildcard applies only when retrieving all settings, not for an individual setting. For details, see Settings. For information about using the manifest file, see Deploying and Retrieving Metadata with the Zip File.

SEE ALSO:

Document

AddressSettings

Represents the configuration of country and state picklists. Use the AddressSettings component type to configure state and country data in your organization so that you can convert text-based values into standard picklist values. To convert your state and country values, from Setup, enter *State and Country Picklists* in the `Quick Find` box, then select **State and Country Picklists**. For more information, see "Let Users Select State and Country from Picklists" in the Salesforce online help.

This type extends the Metadata metadata type and inherits its `fullName` field.

In the package manifest, all organization settings metadata types are accessed using the "Settings" name. See Settings for more details.

Declarative Metadata File Suffix and Directory Location

AddressSettings values are stored in a single file named `Address.settings` in the `settings` directory. The `.settings` files are different from other named components because there is only one settings file for each settings component.

Version

AddressSettings is available in API versions 27.0 and later.

CountriesAndStates

This complex metadata type represents valid definitions of states and countries in picklists.

Note: You can use the Metadata API to edit existing states and countries in state and country picklists. You can't use the Metadata API to create or delete new states or countries.

Field	Field Type	Description
`countries`	Country[]	The countries available in picklists.

Country

This metadata type provides the definition for a country in a picklist.

Field	Field Type	Description
`active`	boolean	Determines whether the value is available in the API. Important: After you enable state and country picklists in your Salesforce organization, you can't set the `active` status to *`false`*.
`integrationValue`	string	A customizable text value that is linked to a state or country code. Integration values for standard states and countries default to the full ISO-standard state and country names. Integration values function similarly to the API names of custom fields and objects. Configuring integration values allows integrations that you set up before enabling state and country picklists to continue to work. Important: If you don't specify integration values before enabling state and country picklists in your organization, records use the default value provided by Salesforce. If you change integration values later, records created or updated from that point on use your edited values.
`isoCode`	string	The ISO-standard code populates this field when you issue a `retrieve()` call. This field is read only in the API but you can edit the label in Setup. You can't edit the `isoCode` of `standard` states and countries.

Field	Field Type	Description
`label`	string	The label is what users see in picklists in Salesforce. This field is read only in the API but you can edit the label in Setup.
`orgDefault`	boolean	Sets a country as the default value for new records in the Salesforce organization.
`standard`	boolean	Standard states and countries are states and countries that are included with Salesforce. You can't edit the `standard` attribute.
`states`	State[]	The states or provinces that are part of the country.
`visible`	boolean	Makes the state or country available to users in Salesforce. States or countries that are `visible` must also be `active`.

State

This metadata type provides the definition for a state in a picklist.

Field	Field Type	Description
`active`	boolean	Determines whether the value is available in the API. Important: After you enable state and country picklists in your Salesforce organization, you can't set the `active` status to *`false`*.
`integrationValue`	string	A customizable text value that is linked to a state or country code. Integration values for standard states and countries default to the full ISO-standard state and country names. Integration values function similarly to the API names of custom fields and objects. Configuring integration values allows integrations that you set up before enabling state and country picklists to continue to work. Important: If you don't specify integration values before enabling state and country picklists in your organization, records use the default value provided by Salesforce. If you change integration values later, records created or updated from that point on use your edited values.
`isoCode`	string	The ISO-standard code populates this field when you issue a `retrieve()` call. This field is read only in the API but you can edit the label in Setup.
`label`	string	The label is what users see in picklists in Salesforce. This field is read only in the API but you can edit the label in Setup.

Field	Field Type	Description
`standard`	boolean	Standard states and countries are states and countries that are included with Salesforce. You can't edit the `standard` attribute.
`visible`	boolean	Makes the state or country available to users in Salesforce. States or countries that are `visible` must also be `active`.

Declarative Metadata Sample Definition

The following is sample XML that configures state and country picklists for the United States and Canada for use in an organization. It also makes the country of Greenland available only in the API. This example is supported in API version 48.0.

```
<?xml version="1.0" encoding="UTF-8"?>
<AddressSettings xmlns="http://soap.sforce.com/2006/04/metadata">
  <countriesAndStates>
    <countries>
      <country>
        <active>true</active>
        <integrationValue>United States</integrationValue>
        <isoCode>US</isoCode>
        <label>United States</label>
        <orgDefault>true</orgDefault>
        <standard>true</standard>
        <states>
          <state>
            <active>true</active>
            <integrationValue>Alabama</integrationValue>
            <isoCode>AL</isoCode>
            <label>Alabama</label>
            <standard>true</standard>
            <visible>true</visible>
          </state>
          <state>
            <active>true</active>
            <integrationValue>Alaska</integrationValue>
            <isoCode>AK</isoCode>
            <label>Alaska</label>
            <standard>true</standard>
            <visible>true</visible>
          </state>
        </states>
        <visible>true</visible>
      </country>
      <country>
        <active>true</active>
        <integrationValue>Canada</integrationValue>
        <isoCode>CA</isoCode>
        <label>Canada</label>
        <orgDefault>false</orgDefault>
        <states>
          <state>
```

```
            <active>true</active>
            <integrationValue>Alberta</integrationValue>
            <isoCode>AB</isoCode>
            <label>Alberta</label>
             <standard>true</standard>
             <visible>true</visible>
          </state>
          <state>
            <active>true</active>
            <integrationValue>British Columbia</integrationValue>
            <isoCode>BC</isoCode>
            <label>British Columbia</label>
            <standard>true</standard>
            <visible>true</visible>
          </state>
        </states>
        <visible>true</visible>
      </country>
      <country>
        <active>true</active>
        <integrationValue>Greenland</integrationValue>
        <isoCode>GL</isoCode>
        <label>Greenland</label>
        <standard>true</standard>
        <visible>false</visible>
      </country>
    </countries>
  </countriesAndStates>
</AddressSettings>
```

Wildcard Support in the Manifest File

The wildcard character `*` (asterisk) in the `package.xml` manifest file doesn't apply to metadata types for feature settings. The wildcard applies only when retrieving all settings, not for an individual setting. For details, see Settings. For information about using the manifest file, see Deploying and Retrieving Metadata with the Zip File.

AnalyticsSettings

Represents Analytics settings in your Salesforce org. Einstein Analytics allows you to explore all of your data quickly and easily by providing AI-powered advanced Analytics, right inside Salesforce. Manage your datasets, query data with Salesforce Analytics Query Language (SAQL), and customize dashboards. You can use these settings to configure which Analytics features are available to users in your org. This type extends the Metadata metadata type and inherits its `fullName` field.

File Suffix and Directory Location

AnalyticsSettings values are stored in the `Analytics.settings` file in the `settings` folder. The `.settings` files are different from other named components because there is only one settings file for each settings component.

Version

AnalyticsSettings components are available in API version 46.0 and later.

Special Access Rules

The AnalyticsSettings metadata type is accessible only in orgs with Einstein Analytics enabled.

Fields

Field Name	Field Type	Description
`alwaysGenPreviews`	boolean	Indicates whether Analytics asset previews are generated (`true`) or not (`false`). Available in API version 47.0 and later.
`analyticsAdoptionMetadata`	boolean	Indicates whether Adoption Analytics metadata collection can be installed via a dataflow in this org (`true`) or not (`false`). Available in API version 47.0 and later.
`canAccessAnalyticsViaAPI`	boolean	Indicates whether Analytics assets can be accessed via the Analytics REST API in this org (`true`) or not (`false`). Available in API version 47.0 and later.
`canAnnotateDashboards`	boolean	Indicates whether the Analytics dashboards Chatter annotation feature is available in this org (`true`) or not (`false`). Available in API version 47.0 and later.
`canEnableLiveMetrics`	boolean	Indicates whether the Data Discovery live model metrics calculation feature is available in this org (`true`) or not (`false`). Available in API version 48.0 and later.
`canEnableSavedView`	boolean	Indicates whether the saved view feature for Analytics dashboards is available in this org (`true`) or not (`false`). Available in API version 47.0 and later.
`canExploreDataConversationally`	boolean	Indicates whether Analytics data can be explored via NLQ (`true`) rather than using strict SAQL statements (`false`). For example, "Show me all accounts that are closed won". Available in API version 47.0 and later.
`canShareAppsWithCommunities`	boolean	Indicates whether Analytics apps can be shared with Communities and their users, outside of the standard Analytics Studio experience (`true`) or not (`false`). Available in API version 47.0 and later.
`canSubscribeDashboardWidgets`	boolean	Indicates whether a user can subscribe to Analytics dashboard widgets in this org (`true`) or not (`false`). Available in API version 47.0 and later.
`canViewThumbnailAssets`	boolean	Indicates whether the thumbnail representations of Analytics lenses and dashboards are viewable (`true`) or not (`false`). Available in API version 47.0 and later.

Field Name	Field Type	Description
enableAnalyticsEncryption	boolean	Indicates whether encryption is on for Analytics in this org (true) or not (false). Available in API version 48.0 and later.
enableAnalyticsSharingEnable	boolean	Indicates whether the Analytics sharing is available in this org (true) or not (false). Available in API version 48.0 and later.
enableAutoCompleteCombo	boolean	Indicates whether using auto-complete when choosing reports and dashboards is available in this org (true) or not (false). Available in API version 47.0 and later.
enableC360GlobalProfileData	boolean	Indicates whether the Customer 360 data validation dashboard connector is available in this org (true) or not (false). Available in API version 48.0 or later.
enableDashboardComponentSnapshot	boolean	Indicates whether posting dashboard component snapshots to feeds that are visible to all users is enabled in this org (true) or not (false). Available in API version 47.0 and later.
enableDashboardFlexiTable	boolean	Indicates whether access is enabled to flexible dashboard tables for all users in this org (true) or not (false). Available in API version 47.0 and later.
enableDashboardToPDFEnable	boolean	Indicates whether a dashboard can be exported to a PDF in this org (true) or not (false). Available in API version 48.0 and later.
enableDataBlending	boolean	Indicates whether the Analytics Explorer data blending feature is available in this org (true) or not (false). Available in API version 48.0 and later.
enableEmailReportsToPortalUsers	boolean	Indicates whether this org allows Classic reports and dashboards to be sent to Portal Users (true) or not (false). Available in API version 47.0 and later.
enableFirebirdEditor	boolean	Indicates whether the Firebird editor is available in this org (true) or not (false). Available in API version 48.0 and later.
enableFloatingReportHeaders	boolean	Indicates whether report results display floating headers when scrolling (true) or not (false). Available in API version 47.0 and later.
enableInsights	boolean	Indicates whether Einstein Analytics is enabled for this org (true) or not (false).

Field Name	Field Type	Description
enableLightningReportBuilder	boolean	Indicates whether the Lightning Report Builder feature can be enabled or disabled on the Setup page in this org (true) or not (false). Available in API version 47.0 and later.
enableLotusNotesImages	boolean	Indicates whether the use of Lotus Notes-friendly images in dashboards and report emails is available in this org (true) or not (false). Available in API version 47.0 and later.
enableMassEnableReportBuilder	boolean	Indicates whether the Report Builder can be enabled for the entire org (true), overriding profile level settings, or not (false). Available in API version 47.0 and later.
enableNewChartsEngine	boolean	Indicates whether the New Charts Engine for reports and dashboards is available in this org (true) or not (false). Available in API version 47.0 and later.
enableNullDimension	boolean	Indicates whether null values are supported as a grouping key value in a SAQL query in this org (true) or not (false). Available in API version 48.0 and later.
enablePowerInsights	boolean	Indicates whether admins are enabled to turn on Power Insights for this org (true) or not (false).
enableQueryLiveConnectors	boolean	Indicates whether querying live connectors is available in this org (true) or not (false). Available in API version 48.0 and later.
enableRemoveFooterForRepDisplay	boolean	Indicates whether the default disclaimer for the report run page and printable view page is removed (true) or not (false) in this org. Available in API version 47.0 and later.
enableRemoveFooterFromRepExp	boolean	Indicates whether the default footer from the exported (csv/excel) report is removed (true) or not (false) in this org. Available in API version 47.0 and later.
enableReportFieldToFieldPref	boolean	Indicates whether the field-to-field filters features in Lightning Experience Reports is available in this org (true) or not (false). Available in API version 47.0. Removed in API version 48.0.
enableReportNotificationsEnable	boolean	Indicates whether the notification feature for Lightning Experience Reports is available in this org (true) or not (false). Available in API version 48.0 and later.

Field Name	Field Type	Description
enableReportUniqueRowCountPref	boolean	Indicates whether the unique row count aggregate feature in Lightning Experience Reports is available in this org (true) or not (false). Available in API version 47.0. Removed in API version 48.0.
enableS1AnalyticsEclairEnable	boolean	Indicates whether EclairNG charts can be enabled for S1 Mobile Analytics in this org (true) or not (false). Available in API version 48.0 and later.
enableSFXJoinedReportsEnable	boolean	Indicates whether the Lightning joined report feature can be enabled or disabled on the Setup page in this org (true) or not (false). Available in API version 47.0 and later.
enableSmartDataDiscovery	boolean	Indicates whether the org admin can enable Einstein Discovery in this org (true) or not (false).
enableUseOldChartsLookAndFeel	boolean	Indicates whether this org allows old charts look and feel for reports and dashboards (true) or not (false). Available in API version 47.0 and later.
enableWaveRecordNavigation	boolean	Indicates whether browser tab navigation for record actions from Analytics is enabled in this org (true) or not (false). Available in API version 48.0 and later.
enableWaveReplication	boolean	Indicates whether replication(extract) for Salesforce objects is enabled in this org (true) instead of SFDC Digest (false).
enableWaveSharingInheritance	boolean	Indicates whether Analytics data can inherit sharing and security settings for their source Salesforce object in this org (true) or not (false).
enableWaveTemplate	boolean	Indicates whether Analytics templates are enabled for this org (true) or not (false).
enableWaveTrendedDatasetCleanup	boolean	Indicates whether this org allows automatic deletion of inactive trended datasets (true) or not (false). Available in API version 47.0 and later.
inheritSharingForNonOpptyObjects	boolean	Indicates whether medium visibility support for Analytics sharing inheritance for all Salesforce objects besides the Opportunity object is available in this org (true) or not (false). Available in API version 48.0 and later.
inheritSharingForOpptyObject	boolean	Indicates whether medium visibility support for Analytics sharing inheritance for the Opportunity object is available in this org (true) or not (false). Available in API version 48.0 and later.

Field Name	Field Type	Description
replaceBlankMeasuresWithNulls	boolean	Indicates whether null values are supported in measures (true) or not (false). If enabled (true), the implicit default value for blank measures is null. Available in API version 48.0 and later.
turnOnTimeZones	boolean	Indicates whether the timezone feature is available in this org (true) or not (false). Available in API version 48.0 and later.

Declarative Metadata Sample Definition

The following is an example of the Analytics.settings file:

```xml
<?xml version="1.0" encoding="UTF-8"?>
<AnalyticsSettings xmlns="http://soap.sforce.com/2006/04/metadata">
    <enableWaveTemplate>true</enableWaveTemplate>
    <enableInsights>true</enableInsights>
    <canAccessAnalyticsViaAPI>true</canAccessAnalyticsViaAPI>
</AnalyticsSettings>
```

Example Package Manifest

The following is an example package manifest used to deploy or retrieve the Analytics settings metadata:

```xml
<?xml version="1.0" encoding="UTF-8"?>
<Package xmlns="http://soap.sforce.com/2006/04/metadata">
    <types>
        <members>Analytics</members>
        <name>Settings</name>
    </types>
    <version>47.0</version>
</Package>
```

Wildcard Support in the Manifest File

The wildcard character * (asterisk) in the package.xml manifest file doesn’t apply to metadata types for feature settings. The wildcard applies only when retrieving all settings, not for an individual setting. For details, see Settings. For information about using the manifest file, see Deploying and Retrieving Metadata with the Zip File.

ApexSettings

Represents Apex-related org settings. This type extends the Metadata metadata type and inherits its fullName field.

In the package manifest, all organization settings metadata types are accessed using the “Settings” name. See Settings for more details.

File Suffix and Directory Location

ApexSettings values are stored in the Apex.settings file in the settings directory of the corresponding package directory. The .settings files are different from other named components, because there is only one settings file for each settings component.

Version

ApexSettings components are available in API version 47.0 and later.

Fields

Field Name	Field Type	Description
`enableAggregateCodeCoverageOnly`	boolean	Indicates whether aggregate (not detailed) totals are tracked for Apex test coverage data (`true`) or not (`false`). The default value is `false`.
`enableApexAccessRightsPref`	boolean	Deprecated.
`enableApexApprovalLockUnlock`	boolean	Indicates whether approval process lock and unlock operations from Apex code are allowed (`true`) or not (`false`). The default value is `false`.
`enableApexCtrlImplicitWithSharingPref`	boolean	Indicates whether the Use `with sharing` for `@AuraEnabled` Apex Controllers with Implicit Sharing critical update is activated (`true`) or not (`false`). For more details, see the Winter '20 Release Notes.
`enableApexPropertyGetterPref`	boolean	Indicates whether the Enforce Access Modifiers on Apex Properties in Lightning Component Markup critical update is activated (`true`) or not (`false`). For more details, see the Winter '20 Release Notes.
`enableAuraApexCtrlAuthUserAccessCheckPref`	boolean	Indicates whether the Restrict Access to `@AuraEnabled` Apex Methods for Authenticated Users Based on User Profile critical update is activated (`true`) or not (`false`). For more details, see the Winter '20 Release Notes.
`enableAuraApexCtrlGuestUserAccessCheckPref`	boolean	Indicates whether the Restrict Access to `@AuraEnabled` Apex Methods for Guest and Portal Users Based on User Profile critical update is activated (`true`) or not (`false`). For more details, see the Winter '20 Release Notes.
`enableCompileOnDeploy`	boolean	Indicates whether Apex code is automatically recompiled (`true`) or not (`false`). When set to `true`, code is recompiled before completing a metadata deployment, change set deployment, package installation, or package upgrade. The default value is `true` for production orgs and `false` for others. Note: This setting cannot be disabled in production orgs.

Field Name	Field Type	Description
enableDisableParallelApexTesting	boolean	Indicates whether Apex tests are serially executed (true) or not (false). The default value is false. Note: Even when parallel testing is enabled by setting this value to false, tests that are run during deployments are always run serially.
enableDoNotEmailDebugLog	boolean	Indicates whether Apex debug log details are suppressed in unhandled exception emails (true) or not (false). The default value is false.
enableGaplessTestAutoNum	boolean	Indicates whether autonumbering gaps are prevented by Apex test executions not incrementing autonumber fields for non-test records (true) or not (false). The default value is true.
enableMngdCtrlActionAccessPref	boolean	Indicates whether the Disable Access to Non-global Apex Controller Methods in Managed Packages critical update is activated (true) or not (false). For more details, see the Winter '20 Release Notes.
enableNonCertifiedApexMdCrud	boolean	Indicates whether Apex classes can access metadata, public or protected, through classes in the Metadata namespace (true) or not (false). The default value is false.
enableSecureNoArgConstructorPref	boolean	Indicates whether Apex type visibility rules are strictly enforced for the Type.newInstance method (true) or not (false). The default value is false. When enabled, regardless of API version, you can instantiate only Apex classes with a no-arguments constructor that is visible to the code running Type.newInstance. Available in API version 48.0 and later.

Declarative Metadata Sample Definition

The following is an example of ApexSettings components.

```
<?xml version="1.0" encoding="UTF-8"?>
<ApexSettings xmlns="http://soap.sforce.com/2006/04/metadata">
   <enableDoNotEmailDebugLog>true</enableDoNotEmailDebugLog>
   <enableDisableParallelApexTesting>true</enableDisableParallelApexTesting>
</ApexSettings>
```

The following is an example package.xml manifest that references the previous definition.

```
<?xml version="1.0" encoding="UTF-8"?>
<Package xmlns="http://soap.sforce.com/2006/04/metadata">
    <types>
        <members>Apex</members>
        <name>Settings</name>
    </types>
    <version>47.0</version>
</Package>
```

Wildcard Support in the Manifest File

The wildcard character * (asterisk) in the package.xml manifest file doesn't apply to metadata types for feature settings. The wildcard applies only when retrieving all settings, not for an individual setting. For details, see Settings. For information about using the manifest file, see Deploying and Retrieving Metadata with the Zip File.

AppExperienceSettings

Represents settings for the app experience.This type extends the Metadata metadata type and inherits its fullName field.

In the package manifest, all organization settings metadata types are accessed using the "Settings" name. See Settings for more details.

File Suffix and Directory Location

AppExperienceSettings values are stored in the .settings file in the settings directory. The .settings files are different from other named components because there is only one settings file for each settings component.

Version

AppExperienceSettings components are available in API version 47.0 and later.

Fields

Field Name	Field Type	Description
doesHideAllAppsInAppLauncher	boolean	If set to false (default), all standard and custom apps show up on the App Launcher. If set to true, the admin must select which standard and custom apps to display on the App Launcher.

Declarative Metadata Sample Definition

The following is an example of an AppExperienceSettings component.

```
<?xml version="1.0" encoding="UTF-8"?>
<AppExperienceSettings xmlns="http://soap.sforce.com/2006/04/metadata">
    <doesHideAllAppsInAppLauncher>false</doesHideAllAppsInAppLauncher>
</AppExperienceSettings>
```

Example Package Manifest

The following is an example package manifest used to deploy or retrieve the AppExperienceSettings metadata for an organization:

```
<?xml version="1.0" encoding="UTF-8"?>
<Package xmlns="http://soap.sforce.com/2006/04/metadata">
    <types>
        <members>AppExperience</members>
        <name>Settings</name>
    </types>
<version>47.0</version>
</Package>
```

Wildcard Support in the Manifest File

The wildcard character `*` (asterisk) in the `package.xml` manifest file doesn't apply to metadata types for feature settings. The wildcard applies only when retrieving all settings, not for an individual setting. For details, see Settings. For information about using the manifest file, see Deploying and Retrieving Metadata with the Zip File.

AutomatedContactsSettings

Represents an org's Einstein Automated Contacts settings. These settings let you find new contacts and opportunity contact roles. This type extends the Metadata metadata type and inherits its `fullName` field.

In the package manifest, all organization settings metadata types are accessed using the "Settings" name. See Settings for more details.

File Suffix and Directory Location

AutomatedContactsSettings values are stored in the `AutomatedContacts.settings` file in the `settings` folder. The `.settings` files are different from other named components because there is only one settings file for each settings component.

Version

AutomatedContactsSettings is available in API versions 48.0 and later.

Fields

Field Name	Field Type	Description
`enableAddContactAutomatically`	boolean	Indicates whether new contacts are automatically added from external email accounts (such as Microsoft and Google) to Salesforce (`true`) or not (`false`). The default value is `false`. `enableAddContactWithSuggestion` must be `true` to use this setting. Note: When this feature is enabled, users do not see new contacts as suggestions. The contacts are added automatically.

Field Name	Field Type	Description
enableAddContactRoleAutomatically	boolean	Indicates whether opportunity contact roles from external accounts are automatically added to Salesforce (`true`) or not (`false`). The default value is `false`. Note: When this feature is enabled, users do not see new contact roles as suggestions. The contact roles are added automatically.
enableAddContactRoleWithSuggestion	boolean	Indicates whether opportunity contact roles from external accounts are suggested as new Salesforce opportunity contact roles (`true`) or not (`false`). The default value is `true`.
enableAddContactWithSuggestion	boolean	Indicates whether new contacts from external accounts (such as Microsoft and Google) are suggested as new Salesforce contacts (`true`) or not (`false`). The default value is `true`. `enableAddContactRoleWithSuggestion` must be `true` to use this setting.

Declarative Metadata Sample Definition

The following is an example of the AutomatedContactsSettings.settings file:

```
<?xml version="1.0" encoding="UTF-8"?>
<AutomatedContactsSettings xmlns="http://soap.sforce.com/2006/04/metadata">
 <enableAddContactAutomatically>true</enableAddContactAutomatically>
 <enableAddContactRoleAutomatically>true</enableAddContactRoleAutomatically>
 <enableAddContactRoleWithSuggestion>true</enableAddContactRoleWithSuggestion>
 <enableAddContactWithSuggestion>true</enableAddContactWithSuggestion>

</AutomatedContactsSettings>
```

Example Package Manifest

The following is an example package manifest used to deploy or retrieve the AutomatedContacts settings metadata:

```
<?xml version="1.0" encoding="UTF-8"?>
<Package xmlns="http://soap.sforce.com/2006/04/metadata">
 <types>
  <members>AutomatedContactsSettings</members>
  <name>Settings</name>
 </types>
 <version>29.0</version>
</Package>
```

Wildcard Support in the Manifest File

The wildcard character `*` (asterisk) in the `package.xml` manifest file doesn't apply to metadata types for feature settings. The wildcard applies only when retrieving all settings, not for an individual setting. For details, see Settings. For information about using the manifest file, see Deploying and Retrieving Metadata with the Zip File.

BotSettings

Represents an organization's Einstein Bot settings, such as whether or not Einstein Bots is enabled. This type extends the Metadata metadata type and inherits its `fullName` field.

File Suffix and Directory Location

Bot components have the suffix `.bot` and are stored in the `bot` folder.

Version

Bot components are available in API version 46.0 and later.

Fields

Field Name	Field Type	Description
`enableBots`	boolean	Indicates whether Einstein Bots is enabled (`true`) or not (`false`).

Declarative Metadata Sample Definition

The following is an example of a BotSetting. This example has been trimmed to make it easier to read.

```
<?xml version="1.0" encoding="UTF-8"?>
<BotSettings xmlns="http://soap.sforce.com/2006/04/metadata">
    <enableBots>true</enableBots>
</BotSettings>
```

The following is an example `package.xml` manifest that references the previous definition.

```
<?xml version="1.0" encoding="UTF-8"?>
<Package xmlns="http://soap.sforce.com/2006/04/metadata">
    <types>
        <members>Bot</members>
        <name>Settings</name>
    </types>
    <version>46.0</version>
</Package>
```

Wildcard Support in the Manifest File

The wildcard character `*` (asterisk) in the `package.xml` manifest file doesn't apply to metadata types for feature settings. The wildcard applies only when retrieving all settings, not for an individual setting. For details, see Settings. For information about using the manifest file, see Deploying and Retrieving Metadata with the Zip File.

BusinessHoursSettings

Represents the metadata used to manage settings for business hours and holidays in entitlements, entitlement templates, campaigns, and cases. This type extends the Metadata metadata type and inherits its `fullName` field.

In the package manifest, all organization settings metadata types are accessed using the “Settings” name. See Settings for more details.

File Suffix and Directory Location

Business hours and holidays settings are stored in a single file named `businessHours.settings` in the `settings` directory. The `.settings` files are different from other named components because there is only one settings file for each settings component.

Version

BusinessHoursSettings is available in API version 29.0 and later.

Fields

Field Name	Field Type	Description
`businessHours`	BusinessHoursEntry[]	Represents the application of business hours to entitlements, entitlement templates, campaigns, and cases.
`holidays`	Holidays[]	Represents a holiday and its usage in `businessHours`.

BusinessHoursEntry

Represents the application of business hours to entitlements, entitlement templates, campaigns, and cases.

Field Name	Field Type	Description
`timeZoneId`	string	The time zone for the time that defines business hours.
`name`	string	Name of the business hours. This name should be unique.
`active`	string	Indicates whether the business hours are active.
`default`	string	Indicates whether the business hours are used as the default business hours.
`mondayStartTime`	string	Start time for the business hours on Monday. Uses the format `HH:mm:ss.SSSZ`.
`mondayEndTime`	string	End time for the business hours on Monday. Uses the format `HH:mm:ss.SSSZ`. The value `00:00:00.000Z` specifies midnight on Monday.
`tuesdayStartTime`	string	Start time for the business hours on Tuesday. Uses the format `HH:mm:ss.SSSZ`.
`tuesdayEndTime`	string	End time for the business hours on Tuesday. Uses the format `HH:mm:ss.SSSZ`. The value `00:00:00.000Z` specifies midnight on Tuesday.
`wednesdayStartTime`	string	Start time for the business hours on Wednesday. Uses the format `HH:mm:ss.SSSZ`.

Field Name	Field Type	Description
`wednesdayEndTime`	string	End time for the business hours on Wednesday. Uses the format `HH:mm:ss.SSSZ`. The value `00:00:00.000Z` specifies midnight on Wednesday.
`thursdayStartTime`	string	Start time for the business hours on Thursday. Uses the format `HH:mm:ss.SSSZ`.
`thursdayEndTime`	string	End time for the business hours on Thursday. Uses the format `HH:mm:ss.SSSZ`. The value `00:00:00.000Z` specifies midnight on Thursday.
`fridayStartTime`	string	Start time for the business hours on Friday. Uses the format `HH:mm:ss.SSSZ`.
`fridayEndTime`	string	End time for the business hours on Friday. Uses the format `HH:mm:ss.SSSZ`. The value `00:00:00.000Z` specifies midnight on Friday.
`saturdayStartTime`	string	Start time for the business hours on Saturday. Uses the format `HH:mm:ss.SSSZ`.
`saturdayEndTime`	string	End time for the business hours on Saturday. Uses the format `HH:mm:ss.SSSZ`. The value `00:00:00.000Z` specifies midnight on Saturday.
`sundayStartTime`	string	Start time for the business hours on Sunday. Uses the format `HH:mm:ss.SSSZ`.
`sundayEndTime`	string	End time for the business hours on Sunday. Uses the format `HH:mm:ss.SSSZ`. The value `00:00:00.000Z` specifies midnight on Sunday.

Holidays

Represents a holiday and its usage in `businessHours`.

Field Name	Field Type	Description
`name`	string	Name of the holiday. This name does not have to be unique.
`description`	string	The description of the holiday.
`isRecurring`	string	Indicates whether the holiday is recurring.
`activityDate`	string	The date of the holiday. Use for non-recurring holidays. Uses the format `HH:mm:ss.SSSZ`.
`recurrenceStartDate`	string	The date the holiday starts recurring. Uses the format `yyyy-mm-dd`.
`recurrenceEndDate`	string	The date the holiday stops recurring. Uses the format `yyyy-mm-dd`. Optional.

Field Name	Field Type	Description
startTime	string	The start time on the date of the holiday. Uses the format HH:mm:ss.SSSZ. startTime and endTime must be both null or both not null. If they are both null, indicates the whole day.
endTime	string	The end time on the date of the holiday. Uses the format HH:mm:ss.SSSZ. startTime and endTime must be both null or both not null. If they are both null, indicates the whole day.
recurrenceType	string	The recurrence type of the holiday. Valid values are: RecursDaily, RecursEveryWeekday, RecursMonthly, RecursMonthlyNth, RecursWeekly, RecursYearly, RecursYealyNth.
recurrenceInterval	string	The interval of weeks, months, or years the holiday recurs.
recurrenceDayOfWeek	string	The day of week the holiday recurs. Valid values: Monday, Tuesday, Wednesday, Thursday, Friday, Saturday, Sunday.
recurrenceDayOfMonth	string	The day of month the holiday recurs. Valid values: integers 1-31.
recurrenceInstance	string	Valid values: First, Second, Third, Fourth, Last. Only used for recurrenceType RecursMonthlyNth and RecursYearlyNth. For example, if the recurenceInstance value is First, the holiday recurs on the first Monday of the month every 3 months.
recurrenceMonthOfYear	string	Valid values: January, February, March, April, May, June, July, August, September, October, November, December.
businessHours	string	The name of the business hours setting that applies to this holiday.

Declarative Metadata Sample Definition

The following is an example businesshours.settings metadata file:

```
<?xml version="1.0" encoding="UTF-8"?>
<BusinessHoursSettings xmlns="http://soap.sforce.com/2006/04/metadata">
    <businessHours>
        <active>true</active>
        <default>true</default>
        <fridayEndTime>00:00:00.000Z</fridayEndTime>
        <fridayStartTime>00:00:00.000Z</fridayStartTime>
        <mondayEndTime>00:00:00.000Z</mondayEndTime>
        <mondayStartTime>00:00:00.000Z</mondayStartTime>
        <name>Default</name>
        <saturdayEndTime>00:00:00.000Z</saturdayEndTime>
        <saturdayStartTime>00:00:00.000Z</saturdayStartTime>
        <sundayEndTime>00:00:00.000Z</sundayEndTime>
        <sundayStartTime>00:00:00.000Z</sundayStartTime>
        <thursdayEndTime>00:00:00.000Z</thursdayEndTime>
        <thursdayStartTime>00:00:00.000Z</thursdayStartTime>
        <timeZoneId>America/Los_Angeles</timeZoneId>
        <tuesdayEndTime>00:00:00.000Z</tuesdayEndTime>
        <tuesdayStartTime>00:00:00.000Z</tuesdayStartTime>
        <wednesdayEndTime>00:00:00.000Z</wednesdayEndTime>
        <wednesdayStartTime>00:00:00.000Z</wednesdayStartTime>
    </businessHours>
    <businessHours>
        <active>true</active>
        <default>false</default>
        <fridayEndTime>00:00:00.000Z</fridayEndTime>
        <fridayStartTime>00:00:00.000Z</fridayStartTime>
        <mondayEndTime>15:00:00.000Z</mondayEndTime>
        <mondayStartTime>09:00:00.000Z</mondayStartTime>
        <name>bh1</name>
        <saturdayEndTime>00:00:00.000Z</saturdayEndTime>
        <saturdayStartTime>00:00:00.000Z</saturdayStartTime>
        <sundayEndTime>00:00:00.000Z</sundayEndTime>
        <sundayStartTime>00:00:00.000Z</sundayStartTime>
        <thursdayEndTime>17:00:00.000Z</thursdayEndTime>
        <thursdayStartTime>10:50:00.000Z</thursdayStartTime>
        <timeZoneId>America/Los_Angeles</timeZoneId>
        <tuesdayEndTime>13:00:00.000Z</tuesdayEndTime>
        <tuesdayStartTime>09:00:00.000Z</tuesdayStartTime>
        <wednesdayEndTime>15:00:00.000Z</wednesdayEndTime>
        <wednesdayStartTime>09:00:00.000Z</wednesdayStartTime>
    </businessHours>
    <holidays>
        <activityDate>2013-09-02</activityDate>
        <businessHours>Default</businessHours>
        <businessHours>bh1</businessHours>
        <isRecurring>false</isRecurring>
        <name>Labor Day</name>
    </holidays>
    <holidays>
        <businessHours>bh1</businessHours>
        <isRecurring>true</isRecurring>
        <name>Thanksgiving</name>
        <recurrenceDayOfMonth>21</recurrenceDayOfMonth>
        <recurrenceMonthOfYear>November</recurrenceMonthOfYear>
        <recurrenceStartDate>2013-11-21</recurrenceStartDate>
        <recurrenceType>RecursYearly</recurrenceType>
    </holidays>
</BusinessHoursSettings>
```

The following is an example `package.xml` manifest that references the BusinessHoursSettings definitions:

```
<?xml version="1.0" encoding="UTF-8"?>
<Package xmlns="http://soap.sforce.com/2006/04/metadata">
    <types>
        <members>BusinessHours</members>
        <name>Settings</name>
    </types>
    <version>29.0</version>
</Package>
```

Wildcard Support in the Manifest File

The wildcard character * (asterisk) in the package.xml manifest file doesn't apply to metadata types for feature settings. The wildcard applies only when retrieving all settings, not for an individual setting. For details, see Settings. For information about using the manifest file, see Deploying and Retrieving Metadata with the Zip File.

CampaignSettings

Represents an org's Campaign Influence settings, which help you understand how your campaigns are affecting your opportunity pipeline. This type extends the Metadata metadata type and inherits its fullName field.

In the package manifest, all organization settings metadata types are accessed using the "Settings" name. See Settings for more details.

File Suffix and Directory Location

CampaignSettings values are stored in the Campaign.settings file in the settings folder. The .settings files are different from other named components because there is only one settings file for each settings component.

Version

CampaignSettings is available in API versions 48.0 and later.

Fields

Field Name	Field Type	Description
enableAutoCampInfluenceDisabled	boolean	Indicates whether Salesforce creates Campaign Influence information (true) or not (false). enableCampaignInfluence2 must be false to use this setting. The default value is false.
enableB2bmaCampaignInfluence2	boolean	Indicates whether your org can access campaign influence models from other systems, such as Pardot (true) or not (false). enableCampaignInfluence2 must be true to use this setting. The default value is false.
enableCampaignHistoryTrackEnabled	boolean	This read-only field is reserved for system use.
enableCampaignInfluence2	boolean	Indicates whether Customizable Campaign Influence is enabled (true) or not (false). When true, Campaign Influence 1.0 is hidden from users and is no longer active. The default value is true.
enableCampaignMemberTWCF	boolean	This read-only field is reserved for system use.
enableSuppressNoValueCI2	boolean	This read-only field is reserved for system use.

Declarative Metadata Sample Definition

The following is an example of the Campaign.settings file:

```
<?xml version="1.0" encoding="UTF-8"?>
<CampaignSettings xmlns="http://soap.sforce.com/2006/04/metadata">
 <enableAutoCampInfluenceDisabled>false</enableAutoCampInfluenceDisabled>
 <enableB2bmaCampaignInfluence2>true</enableB2bmaCampaignInfluence2>
 <enableCampaignInfluence2>true</enableCampaignInfluence2>
</CampaignSettings>
```

Example Package Manifest

The following is an example package manifest used to deploy or retrieve the Campaign settings metadata:

```
<?xml version="1.0" encoding="UTF-8"?>
<Package xmlns="http://soap.sforce.com/2006/04/metadata">
 <types>
  <members>Campaign</members>
  <name>Settings</name>
 </types>
 <version>29.0</version>
</Package>
```

Wildcard Support in the Manifest File

The wildcard character `*` (asterisk) in the `package.xml` manifest file doesn’t apply to metadata types for feature settings. The wildcard applies only when retrieving all settings, not for an individual setting. For details, see Settings. For information about using the manifest file, see Deploying and Retrieving Metadata with the Zip File.

CaseClassificationSettings

Represents the Einstein Case Classification settings for an org. This type extends the Metadata metadata type and inherits its `fullName` field.

In the package manifest, all organization settings metadata types are accessed using the “Settings” name. See Settings for more details.

File Suffix and Directory Location

CaseClassificationSettings values are stored in the `CaseClassification.settings` file in the `settings` folder. The `.settings` files are different from other named components because there is only one settings file for each settings component.

Version

CaseClassificationSettings components are available in API version 47.0 and later.

Fields

Field Name	Field Type	Description
`caseClassificationRecommendations`	boolean	Indicates whether Einstein Case Classification is enabled in your org. The default value is `false`.
`reRunAttributeBasedRules`	boolean	If `true`, attribute setup for skills-based routing rules are run after Einstein Case Classification automatically updates field values. The default value is `false`.
`runAssignmentRules`	boolean	If `true`, assignment rules are run after Einstein Case Classification automatically updates field values. The default value is `false`.

Declarative Metadata Sample Definition

The following is an example of a CaseClassificationSettings component.

```
<?xml version="1.0" encoding="UTF-8"?>
<CaseClassificationSettings xmlns="http://soap.sforce.com/2006/04/metadata">
    <caseClassificationRecommendations>true</caseClassificationRecommendations>
    <reRunAttributeBasedRules>true</reRunAttributeBasedRules>
    <runAssignmentRules>true</runAssignmentRules>
</CaseClassificationSettings>
```

The following is an example `package.xml` that references the previous definition.

```
<?xml version="1.0" encoding="UTF-8"?>
<Package xmlns="http://soap.sforce.com/2006/04/metadata">
    <types>
        <members>CaseClassification</members>
        <name>Settings</name>
    </types>
    <version>47.0</version>
</Package>
```

CaseSettings

Represents an organization's case settings, such as the default case owner, which case-related features are enabled, and which Classic email templates are used for various case activities. This type extends the Metadata metadata type and inherits its `fullName` field.

In the package manifest, all organization settings metadata types are accessed using the "Settings" name. See Settings for more details.

File Suffix and Directory Location

CaseSettings values are stored in the `Case.settings` file in the `settings` directory. The `.settings` files are different from other named components because there is only one settings file for each settings component.

Version

CaseSettings is available in API version 27.0 and later.

Fields

Field Name	Field Type	Description
caseAssignNotificationTemplate	string	Specifies the email template used for case assignment notifications. The format must be folderName/emailTemplateName. Lightning email templates aren't packageable. We recommend using a Classic email template.
caseAutoProcUser	boolean	Indicates whether to create an automated response record after a customer's initial email (true) or not (false).
caseCloseNotificationTemplate	string	Specifies the email template used for case close notifications. The format must be folderName/emailTemplateName. Lightning email templates aren't packageable. We recommend using a Classic email template.
caseCommentNotificationTemplate	string	Specifies the email template used for case comment notifications. The format must be folderName/emailTemplateName. Lightning email templates aren't packageable. We recommend using a Classic email template.
caseCreateNotificationTemplate	string	Specifies the email template used for case create notifications. The format must be folderName/emailTemplateName. Lightning email templates aren't packageable. We recommend using a Classic email template.
caseFeedItemSettings	FeedItemSettings[]	Specifies the settings for feed items in feed-based case page layouts. This field is available in API version 32.0 and later.
caseFeedReadUnreadLtng	boolean	Indicates whether unread feed items are shown in bold in Lightning Experience (true) or not (false).
caseMergeinLightning	boolean	Indicates whether Case Merge is enabled in Lightning Experience (true) or not (false).
closeCaseThroughStatusChange	boolean	Indicates whether Closed is included in the Case Status field on case edit pages (true) or not (false).
defaultCaseFeedLayoutOn	boolean	Indicates whether the default Case Feed layout is used in the org (true) or not (false).
defaultCaseOwner	string	Specifies the default owner of a case when assignment rules fail to locate an owner.
defaultCaseOwnerType	string	Specifies whether the default case owner is a user or a queue.

Field Name	Field Type	Description
`defaultCaseUser`	string	Specifies the user listed in the Case History related list for automated case changes from: • Assignment rules • Escalation rules • On-Demand Email-to-Case • Cases logged in the Self-Service portal Lightning email templates aren't packageable. We recommend using a Classic email template.
`emailActionDefaultsHandlerClass`	string	Use this Apex class name to provide default values for the email action.
`emailToCase`	EmailToCaseSettings	The organization's Email-to-Case settings.
`enableCaseFeed`	boolean	Indicates whether Case Feed is enabled (`true`) or not (`false`).
`enableCollapseEmailThread`	boolean	Indicates whether earlier messages in an email thread are removed from email feed items (`true`) or not (`false`). Available in API version 47.0 and later.
`enableDraftEmails`	boolean	Indicates whether draft emails are enabled (`true`) or not (`false`). Enabling email drafts requires that Case Feed and Email-to-Case are also enabled.
`enableEarlyEscalationRuleTriggers`	boolean	Indicates whether early triggers on escalation rules are enabled (`true`) or not (`false`).
`enableEmailActionDefaultsHandler`	boolean	Indicates whether the Email Action Default Handler setting is enabled (`true`) or not (`false`). Use this setting to select an Apex class to load a default template or to specify the default target fields for the email action.
`enableEmailContactOnCasePost`	boolean	If `true`, the case contact is notified by email when someone makes an externally visible post on a community case.
`enableEscalateQfiToCaseInternal`	boolean	If `true`, moderators can create cases from Question feed items in Chatter in your organization.
`enableEscalateQfiToCaseNetworks`	boolean	If `true`, moderators can create cases from Question feed items in Chatter in all communities where Chatter Questions is enabled.
`enableExtNetworksCaseFeedEnabled`	boolean	If `true`, community members can see case-related emails, comments, and updates in the case feed.

Field Name	Field Type	Description
`enableNewEmailDefaultTemplate`	boolean	Indicates whether default email templates are enabled (`true`) or not (`false`). Default email templates are available only if draft emails are enabled. Lightning email templates aren't packageable. We recommend using a Classic email template.
`enableSuggestedArticlesApplication`	boolean	Indicates whether the Suggested Articles list appears on case pages. (`true`) or not (`false`). Is only valid if `enableSuggestedSolutions=false`.
`enableSuggestedArticlesCustomerPortal`	boolean	Indicates whether the Suggested Articles list appears on customer portal pages (`true`) or not (`false`). Is only valid if `enableSuggestedSolutions=false`.
`enableSuggestedArticlesPartnerPortal`	boolean	Indicates whether the Suggested Articles list appears on partner portal pages (`true`) or not (`false`). Is only valid if `enableSuggestedSolutions=false`.
`enableSuggestedSolutions`	boolean	Indicates whether the View Suggested Solutions or Find Articles button appears on case detail pages (`true`) or not (`false`). Is only valid if `enableSuggestedArticlesApplication`, `enableSuggestedArticlesCustomerPortal`, and `enableSuggestedArticlesPartnerPortal=false`.
`escalateCaseBefore`	boolean	Indicates whether early triggers are enabled to escalate a case (`true`) or not (`false`).
`genericMessageEnabled`	boolean	Indicates whether generic messages are enabled (`true`) or not (`false`).
`keepCaseMergeRecords`	boolean	If `true`, duplicate cases are not deleted after a case merge.
`keepRecordTypeOnAssignmentRule`	boolean	Indicates whether, when applying assignment rules to manually created records, to keep the existing record type (`true`) or to override the existing record type with the assignee's default record type (`false`).
`newEmailDefaultTemplateClass`	string	Specifies the Apex class that defines the default email template for new email messages in Case Feed. This field appears only when `enableNewEmailDefaultTemplate=true`.Lightning email templates aren't packageable. We recommend using a Classic email template.
`notifyContactOnCaseComment`	boolean	Indicates whether contacts who are not members of your Self-Service portal can be notified when a new comment is added to a case (`true`) or not (`false`).

Field Name	Field Type	Description
`notifyDefaultCaseOwner`	boolean	Indicates whether the default case owner is notified when assigned a new case (`true`) or not (`false`).
`notifyOwnerOnCaseComment`	boolean	Indicates whether the case owner is notified when a comment is added to a case (`true`) or not (`false`).
`notifyOwnerOnCaseOwnerChange`	boolean	Indicates whether the `Send Notification Email` checkbox on cases is automatically selected when users change a case owner to another user (`true`).
`predictiveSupportEnabled`	boolean	Indicates whether predictive support is enabled (`true`) or not (`false`).
`showEmailAttachmentsInCaseAttachmentsRL`	boolean	Indicates whether the case Attachments related list shows email attachments. If `true`, the page displays an email icon next to each attachment from an email in the Attachments related list for cases. The related list's list view also includes a Source column that identifies the attachment's origin. If `false`, email attachments aren't displayed in the Attachments related list for cases. This field is available in API version 40.0 and later.
`showFewerCloseActions`	boolean	Indicates whether the **Save & Close** button on case edit pages and the **Cls** link on Cases related lists are hidden (`true`) or shown (`false`).
`systemUserEmail`	string	Specifies the email address used when the default case user is the system user.
`useSystemEmailAddress`	boolean	Indicates whether case comment, case attachment, and case assignment email notifications are sent from a system address (`true`) or whether case notifications appear to be sent from the user or contact updating the case (`false`).
`useSystemUserAsDefaultCaseUser`	boolean	Indicates whether the system user is used as the automated case user (`true`) or not (`false`). If false, then you must specify a value for the `defaultCaseUser` field.
`visibleInCssCheckbox`	boolean	Sets the default visibility of a case as indicated by the Visible in CSS option on the case edit page. If `false`, the case is visible in CSS by default. If `true`, CSS visibility is off.
`webToCase`	WebToCaseSettings	The organization's Web-to-Case settings.

EmailToCaseSettings

Represents an organization's Email-to-Case settings.

Fields

Field Name	Field Type	Description
enableEmailToCase	boolean	Indicates whether Email-to-Case is enabled (true) or not (false). Note: once Email-to-Case is enabled, it can't be disabled.
enableE2CAttachmentAsFile	boolean	Indicates whether to save attachments sent using Email-to-Case as Salesforce Files (true) or not (false).
enableE2CSourceTracking	boolean	Indicates whether Set Case Source to Email is enabled (true) or not (false). After you enable this setting, the **Case Source** field is updated to Email for all cases that originate from Email-to-Case. Associated emails are marked as Read when the agent opens the case.
enableHtmlEmail	boolean	Indicates whether HTML email is enabled (true) or not (false).
enableOnDemandEmailToCase	boolean	Indicates whether On-Demand Email-to-Case is enabled (true) or not (false).
enableThreadIDInBody	boolean	Indicates whether the Thread ID for a case is inserted in the body of an email (true) or not (false).
enableThreadIDInSubject	boolean	Indicates whether the Thread ID for a case is inserted in the subject line of an email (true) or not (false).
notifyOwnerOnNewCaseEmail	boolean	Indicates whether the owner of a case receives a notification when a new email related to the case is received (true) or not (false).
overEmailLimitAction	EmailToCaseOnFailureActionType (enumeration of type string)	Specifies what happens to email messages received after an organization exceeds its daily Email-to-Case limits. Valid values are: • Bounce • Discard • Requeue
preQuoteSignature	boolean	Indicates whether the user signature is inserted after the reply but before the email thread in an outbound email (true) or at the end of the email (false).
routingAddresses	EmailToCaseRoutingAddress[]	The organization's Email-to-Case routing address settings.
unauthorizedSenderAction	EmailToCaseOnFailureActionType (enumeration of type string)	Specifies what happens to email messages received from invalid senders. Valid values are: • Bounce • Discard

EmailToCaseRoutingAddress

Represents an organization’s Email-to-Case routing address.

Fields

Field Name	Field Type	Description
addressType	EmailToCaseRoutingAddressType (enumeration of type string)	Specifies the type of Email-to-Case routing address. Valid values are: • EmailToCase—A routing address used with Email-to-Case or On-Demand Email-to-Case. • Outlook—A routing address used with Salesforce for Outlook to create cases from Outlook. Requires that On-Demand Email-to-Case is enabled.
authorizedSenders	string	Specifies the email addresses or domains from which On-Demand Email-to-Case can receive email. Include multiple entries in a comma-separated list.
caseOrigin	string	Specifies the default case origin for cases created through this routing address.
caseOwner	string	Specifies the default owner of cases created through this routing address. The case owner can be either a user or a queue. Specify the case owner using a Salesforce username. Specifying a case owner here in the routing address settings value of defaultCaseOwner in CaseSettings.
caseOwnerType	string	Specifies whether the default case owner is a user or a queue.
casePriority	string	Specifies the default case priority for cases created through this routing address.
createTask	boolean	Indicates whether a task is automatically assigned to the case owner when a case is created through an email (true) or not (false).
emailAddress	string	Specifies the email address used to route email messages that are submitted as cases.
emailServicesAddress	string	Specifies the Salesforce-generated routing address used for setting up Email-to-Case forwarding. This field value is read-only and can't be modified.
isVerified	boolean	Indicates whether the customer has verified the routing address (typically by clicking a confirmation email). This field value is read-only and can't be modified.
routingName	string	Specifies the name of the Email-to-Case routing address.

Field Name	Field Type	Description
`saveEmailHeaders`	boolean	Indicates whether email routing and envelope information are saved (`true`) or not (`false`).
`taskStatus`	string	Specifies the default status on tasks automatically assigned to the case owner when email is submitted as a case. Only applies if `createTask` is set to `true`.

FeedItemSettings

Represents an organization's feed item settings. Available in API version 32.0 and later.

Field Name	Field Type	Description
`characterLimit`	int	Specifies the maximum number of characters displayed for each feed item.
`collapseThread`	boolean	Removed. Indicates whether earlier messages in an email thread are removed from email feed items (`true`) or not (`false`). Available in API version 27.0 to 46.0.
`displayFormat`	FeedItemDisplayFormat (enumeration of type string)	Indicates how email feed items are displayed. Valid values are: • `Default`—Blank lines in email feed items are displayed. • `HideBlankLines`—Blank lines in email feed items are not displayed.
`feedItemType`	FeedItemType (enumeration of type string)	The type of feed item to which the settings apply. For `FeedItemSettings`, the only valid `feedItemType` value is `EmailMessageEvent`.

WebToCaseSettings

Represents an organization's Web-to-Case settings.

Fields

Field Name	Field Type	Description
`caseOrigin`	string	Specifies the default case origin for cases created through this web form. Only applies if `enableWebToCase` is set to `true`.
`defaultResponseTemplate`	string	Specifies the default template used for email responses to cases submitted through a Self-Service portal. Only applies if `enableWebToCase` is set to `true`. Lightning email templates aren't packageable. We recommend using a Classic email template.

Field Name	Field Type	Description
`enableWebToCase`	boolean	Indicates whether Web-to-Case is enabled (`true`) or not (`false`).

Declarative Metadata Sample Definition

This code sample is an example of a case settings file.

```
<?xml version="1.0" encoding="UTF-8"?>
<CaseSettings xmlns="http://soap.sforce.com/2006/04/metadata">
    <caseAssignNotificationTemplate>
        unfiled$public/SupportCaseAssignmentNotification
    </caseAssignNotificationTemplate>
    <caseCloseNotificationTemplate>
        unfiled$public/SupportCaseCloseNotification
    </caseCloseNotificationTemplate>
    <caseCommentNotificationTemplate>
        unfiled$public/SupportCaseCommentNotification
    </caseCommentNotificationTemplate>
    <caseCreateNotificationTemplate>
        unfiled$public/SupportCaseCreateNotification
    </caseCreateNotificationTemplate>
    <closeCaseThroughStatusChange>true</closeCaseThroughStatusChange>
    <defaultCaseOwner>admin@acme.com</defaultCaseOwner>
    <defaultCaseOwnerType>User</defaultCaseOwnerType>
    <defaultCaseUser>admin@acme.com</defaultCaseUser>
    <emailToCase>
        <enableEmailToCase>true</enableEmailToCase>
        <enableHtmlEmail>false</enableHtmlEmail>
        <enableOnDemandEmailToCase>true</enableOnDemandEmailToCase>
        <enableThreadIDInBody>true</enableThreadIDInBody>
        <enableThreadIDInSubject>true</enableThreadIDInSubject>
        <notifyOwnerOnNewCaseEmail>false</notifyOwnerOnNewCaseEmail>
        <overEmailLimitAction>Bounce</overEmailLimitAction>
        <preQuoteSignature>true</preQuoteSignature>
        <routingAddresses>
            <addressType>EmailToCase</addressType>
            <authorizedSenders>user@acme.com</authorizedSenders>
            <caseOrigin>Email</caseOrigin>
            <casePriority>Medium</casePriority>
            <createTask>true</createTask>
            <emailAddress>support@acme.com</emailAddress>
            <routingName>EmailToCaseRoutingAddress1</routingName>
            <saveEmailHeaders>true</saveEmailHeaders>
            <taskStatus>Not Started</taskStatus>
        </routingAddresses>
        <routingAddresses>
            <addressType>Outlook</addressType>
            <authorizedSenders>user@acme.com</authorizedSenders>
            <caseOrigin>Email</caseOrigin>
            <caseOwner>admin@acme.com</caseOwner>
            <caseOwnerType>User</caseOwnerType>
            <casePriority>High</casePriority>
            <routingName>OutlookRoutingAddress1</routingName>
        </routingAddresses>
        <unauthorizedSenderAction>Discard</unauthorizedSenderAction>
    </emailToCase>
    <enableCaseFeed>true</enableCaseFeed>
    <enableDraftEmails>true</enableDraftEmails>
    <enableEarlyEscalationRuleTriggers>true</enableEarlyEscalationRuleTriggers>
    <enableNewEmailDefaultTemplate>true</enableNewEmailDefaultTemplate>
    <enableSuggestedArticlesApplication>true</enableSuggestedArticlesApplication>
    <enableSuggestedArticlesCustomerPortal>true</enableSuggestedArticlesCustomerPortal>
    <enableSuggestedArticlesPartnerPortal>false</enableSuggestedArticlesPartnerPortal>
    <enableSuggestedSolutions>false</enableSuggestedSolutions>
    <keepRecordTypeOnAssignmentRule>true</keepRecordTypeOnAssignmentRule>
    <newEmailDefaultTemplateClass>CaseTemplateController</newEmailDefaultTemplateClass>
    <notifyContactOnCaseComment>true</notifyContactOnCaseComment>
    <notifyDefaultCaseOwner>true</notifyDefaultCaseOwner>
    <notifyOwnerOnCaseComment>true</notifyOwnerOnCaseComment>
    <notifyOwnerOnCaseOwnerChange>false</notifyOwnerOnCaseOwnerChange>
    <showFewerCloseActions>false</showFewerCloseActions>
    <useSystemEmailAddress>true</useSystemEmailAddress>
    <webToCase>
        <caseOrigin>Web</caseOrigin>
       <defaultResponseTemplate>unfiled$public/SupportCaseResponse</defaultResponseTemplate>

        <enableWebToCase>true</enableWebToCase>
    </webToCase>
</CaseSettings>
```

The following is an example `package.xml` that references the previous definition.

```
<?xml version="1.0" encoding="UTF-8"?>
<Package xmlns="http://soap.sforce.com/2006/04/metadata">
    <types>
        <members>Case</members>
        <name>Settings</name>
    </types>
    <version>47.0</version>
</Package>
```

Wildcard Support in the Manifest File

The wildcard character `*` (asterisk) in the `package.xml` manifest file doesn't apply to metadata types for feature settings. The wildcard applies only when retrieving all settings, not for an individual setting. For details, see Settings. For information about using the manifest file, see Deploying and Retrieving Metadata with the Zip File.

ChatterAnswersSettings

Represents the metadata used to manage settings for Chatter Answers.

In the package manifest, all organization settings metadata types are accessed using the "Settings" name. See Settings for more details.

File Suffix and Directory Location

Chatter Answers settings are stored in a single file named `ChatterAnswers.settings` in the `settings` directory. The `.settings` files are different from other named components because there is only one settings file for each settings component.

Version

ChatterAnswersSettings is available in API version 27.0 and later.

Fields

Field Name	Field Type	Description
`emailFollowersOnBestAnswer`	boolean	Indicates whether users are notified when a best answer is selected for a question that they're following (`true`) or not (`false`).
`emailFollowersOnReply`	boolean	Indicates whether users are notified when other users reply to questions they're following (`true`) or not (`false`).
`emailOwnerOnPrivateReply`	boolean	Indicates whether users are notified when customer support responds to their questions privately (`true`) or not (`false`).
`emailOwnerOnReply`	boolean	Indicates whether users are notified when other users reply to their questions (`true`) or not (`false`).
`enableAnswerViaEmail`	boolean	Indicates whether users can post answers by replying to email notifications (`true`) or not (`false`). This field is available in API version 29.0 and later.
`enableChatterAnswers`	boolean	Indicates whether Chatter Answers is enabled in the organization (`true`) or not (`false`).
`enableFacebookSSO`	boolean	Indicates whether users sign in to your Chatter Answers communities with their Facebook logins (`true`) or not (`false`). To enable this feature, you must define and enable a Facebook authentication provider in your organization's security controls and enable Auth Providers in your organization.
`enableInlinePublisher`	boolean	Indicates whether users can filter search results by articles or questions before they post a question to any of your Chatter Answers communities (`true`) or not (`false`). Also, adds `Title` and `Body` fields to questions for easier text input and scanning. This field is available in API version 29.0 and later.
`enableReputation`	boolean	Indicates whether reputations display for users as hover text on their profile pictures (`true`) or not (`false`). Reputation is enabled across all zones. To enable the reputation setting, you must enable Reputation in your organization.
`enableRichTextEditor`	boolean	Indicates whether the rich text editor is enabled for users to format text and upload images when posting questions (`true`) or not (`false`). To enable rich text editor, you must enable Optimize Question Flow.

Field Name	Field Type	Description
facebookAuthProvider	string	The name of an existing Facebook authentication provider. You must choose a Facebook authentication provider to implement Facebook Single Sign On for your Chatter Answers communities.
showInPortals	boolean	Indicates whether Chatter Answers can be added as a tab to your Customer portal or partner portal (`true`) or not (`false`).

Declarative Metadata Sample Definition

The following is an example `chatteranswers.settings` metadata file:

```
<?xml version="1.0" encoding="UTF-8"?>
<ChatterAnswersSettings xmlns="http://soap.sforce.com/2006/04/metadata">
    <emailFollowersOnBestAnswer>true</emailFollowersOnBestAnswer>
    <emailFollowersOnReply>true</emailFollowersOnReply>
    <emailOwnerOnPrivateReply>true</emailOwnerOnPrivateReply>
    <emailOwnerOnReply>true</emailOwnerOnReply>
    <enableChatterAnswers>true</enableChatterAnswers>
    <enableFacebookSSO>true</enableFacebookSSO>
    <enableInlinePublisher>true</enableInlinePublisher>
    <enableReputation>true</enableReputation>
    <enableRichTextEditor>true</enableRichTextEditor>
    <facebookAuthProvider>FacebookAuthProvider</facebookAuthProvider>
    <showInPortals>true</showInPortals>
</ChatterAnswersSettings>
```

The following is an example `package.xml` manifest that references the ChatterAnswersSettings definitions:

```
<?xml version="1.0" encoding="UTF-8"?>
<Package xmlns="http://soap.sforce.com/2006/04/metadata">
    <types>
        <members>ChatterAnswers</members>
        <name>Settings</name>
    </types>
    <version>29.0</version>
</Package>
```

Wildcard Support in the Manifest File

The wildcard character `*` (asterisk) in the `package.xml` manifest file doesn't apply to metadata types for feature settings. The wildcard applies only when retrieving all settings, not for an individual setting. For details, see Settings. For information about using the manifest file, see Deploying and Retrieving Metadata with the Zip File.

ChatterEmailMDSettings

Represents an org's settings for Chatter email when Chatter is enabled. This type extends the Metadata metadata type and inherits its `fullName` field.

In the package manifest, all organization settings metadata types are accessed using the "Settings" name. See Settings for more details.

File Suffix and Directory Location

The ChatterEmailMDSettings component appears in the `ChatterEmailsMD.settings` file, and is stored in the `settings` folder. The `.settings` files are different from other named components because there is only one settings file for each settings component.

Version

ChatterEmailMDSettings components are available in API version 47.0 and later.

Fields

Field Name	Field Type	Description
`enableChatterDigestEmailsApiOnly`	boolean	Indicates whether Chatter digests can be sent via the API, rather than according to the regular schedule, for your org (`true`) or not (`false`).
`enableChatterEmailAttachment`	boolean	Indicates whether attachments can be included on posts to chatter feeds via email replies (`true`) or not (`false`).
`enableCollaborationEmail`	boolean	Indicates whether collaboration email notifications can be sent (`true`) or not (`false`).
`enableDisplayAppDownloadBadges`	boolean	Indicates whether iOS and Android app download badges display in Chatter notifications (`true`) or not (`false`).
`enableEmailReplyToChatter`	boolean	Indicates whether users can reply to chatter notifications through an email response (`true`) or not (`false`).
`enableEmailToChatter`	boolean	Indicates whether users can post to chatter feeds via email (`true`) or not (`false`).

Declarative Metadata Sample Definition

The following is an example of a `chatteremailmd.settings` metadata file.

```xml
<?xml version="1.0" encoding="UTF-8"?>
<ChatterEmailsMDSettings xmlns="http://soap.sforce.com/2006/04/metadata">
    <enableChatterDigestEmailsApiOnly>false</enableChatterDigestEmailsApiOnly>
    <enableChatterEmailAttachment>false</enableChatterEmailAttachment>
    <enableCollaborationEmail>true</enableCollaborationEmail>
    <enableDisplayAppDownloadBadges>true</enableDisplayAppDownloadBadges>
    <enableEmailReplyToChatter>false</enableEmailReplyToChatter>
    <enableEmailToChatter>true</enableEmailToChatter>
</ChatterEmailsMDSettings>
```

Wildcard Support in the Manifest File

The wildcard character `*` (asterisk) in the `package.xml` manifest file doesn't apply to metadata types for feature settings. The wildcard applies only when retrieving all settings, not for an individual setting. For details, see Settings. For information about using the manifest file, see Deploying and Retrieving Metadata with the Zip File.

ChatterSettings

Represents an org's settings for their Chatter instance when Chatter is enabled for the org. This type extends the Metadata metadata type and inherits its `fullName` field.

In the package manifest, all organization settings metadata types are accessed using the "Settings" name. See Settings for more details.

File Suffix and Directory Location

The ChatterSettings component appears in the `Chatter.settings` file, and is stored in the `settings` folder. The `.settings` files are different from other named components because there is only one settings file for each settings component.

Version

ChatterSettings components are available in API version 47.0 and later.

Fields

Field Name	Field Type	Description
`allowChatterGroupArchiving`	boolean	Indicates whether manual and automatic group archiving are allowed on all Chatter groups (`true`) or aren't allowed (`false`). In Setup, `allowChatterGroupArchiving` equates to the Chatter setting **Allow Group Archiving**.
`allowRecordsInChatterGroup`	boolean	Indicates whether records can be associated with groups (`true`), or not (`false`). Setting this field to `false` doesn't delete existing record data from groups. In Setup, `allowRecordsInChatterGroup` equates to the Chatter setting **Allow Records in Groups**.
`allowSharingInChatterGroup`	boolean	Removed. The setting of this field has no effect on the org. Available in API version 47.0 only.
`enableApprovalRequest`	boolean	Indicates whether Approvals in Chatter are enabled for the org. When the value is `true`, users see approval requests as posts in Chatter feeds. Users can update their own Chatter feeds settings to opt out of receiving approval requests as Chatter posts. When the value is `false`, approval requests aren't posted to Chatter. By default, the value is `false`. In Setup, `enableApprovalRequest` equates to the Chatter setting **Allow Approvals**.
`enableCaseFeedRelativeTimestamps`	boolean	In Case feeds, indicates whether to use relative (`true`) or absolute (`false`) date and time stamp formats on Case feed items. When the value is `true`, Case feed items show a relative time stamp (for example, `10m ago`). When the value is `false`, Case feed items show an absolute time stamp (for example, `January 7, 2020 at 12:15PM`). When you change this setting, all time stamps in Case feeds

Field Name	Field Type	Description
		reflect that change. The default value is `true`. This field is available in API version 48.0 and later. In Setup, `enableCaseFeedRelativeTimestamps` equates to the Chatter setting **Show relative timestamp**.
`enableChatter`	boolean	Indicates whether Chatter is enabled for your org (`true`) or not (`false`).
`enableChatterEmoticons`	boolean	Indicates whether the automatic conversion of text characters, such as :), into a graphic emoticon is allowed in Chatter (`true`) or isn't allowed (`false`). In Setup, `enableChatter` equates to the Chatter setting **Allow Emoticons**.
`enableFeedEdit`	boolean	Indicates whether qualified users can edit feed posts and comments (`true`) or not (`false`). Qualified users include: • The author of the post or comment • The person who owns the record that was posted to or commented on • The Chatter or community moderator In Setup, `enableFeedEdit` equates to the Chatter setting **Allow users to edit posts and comments**.
`enableFeedPinning`	boolean	Indicates whether to allow the pinning of posts in a feed (true) or not (false). When set to true: • Authorized users can pin posts to the top of the feed. • The feed supports up to three pinned posts. • Pinned posts stay pinned until they're unpinned. After post pinning is enabled, authorized users include admins and group owners and managers. Admins can also assign post pinning permission through permission sets or user profiles. In Setup, `enableFeedPinning` equates to the Chatter setting **Allow post pinning**.
`enableFeedsDraftPosts`	boolean	Indicates whether draft posts are automatically saved every seven seconds (`true`) or not (`false`). When set to `true`: • Adds the My Drafts feed to the Chatter tab • Saves draft posts automatically every seven seconds • Makes drafts available in the My Drafts feed When the user posts the entry, the draft is automatically removed from the My Drafts feed. In Setup, `enableFeedsDraftPosts` equates to the Chatter setting **Allow draft posts**.

Field Name	Field Type	Description
`enableFeedsRichText`	boolean	Indicates whether to use the Rich Text Editor in the Chatter Publisher (`true`) or not (`false`). The rich text editor supports text formats, inline images, hyperlinks, and, when enabled for the org, code snippets. In Setup, `enableFeedsRichText` equates to the Chatter setting **Allow users to compose rich text posts**.
`enableInviteCsnUsers`	boolean	Indicates whether licensed users can invite customers to private groups that the licensed user owns or manages (`true`) or not (`false`). Licensed users can invite customers who are from outside org email domains. Invited customers can see information only in the groups they're invited to. They can interact only with members of those groups. In Setup, `enableInviteCsnUsers` equates to the Chatter setting **Allow customer invitations**.
`enableOutOfOfficeEnabledPref`	boolean	Indicates whether to add an Out of Office setting to a user profile page (`true`), or to omit it (`false`). When set to `true`, adds a control to user profile pages for setting a personal out-of-office message. In Setup, `enableOutOfOfficeEnabledPref` equates to the Chatter setting **Users can set Out of Office message**.
`enableRichLinkPreviewsInFeed`	boolean	Indicates whether to convert links in posts into embedded videos, images, and article previews (`true`) or not to convert the links (`false`). In Setup, `enableRichLinkPreviewsInFeed` equates to the Chatter setting **Allow Rich Link Previews**.
`enableTodayRecsInFeed`	boolean	Indicates whether to allow the posting of recommendations for using the Salesforce Today app in users' feeds (`true`) or not (`false`). When set to `true`, automatically posts recommendations for using the Salesforce Today app in users' feeds. In Setup, `enableTodayRecsInFeed` equates to the Chatter setting **Allow Today Recommendations**.
`unlistedGroupsEnabled`	boolean	Indicates whether to allow the creation of unlisted groups (`true`) or to prevent their creation (`false`). When set to `true`, users can create unlisted groups. Unlisted groups do not appear on the Groups list page. Membership in unlisted groups is by invitation only. In Setup, `unlistedGroupsEnabled` equates to the Chatter setting **Enable Unlisted Groups**.

Declarative Metadata Sample Definition

The following is an example of a Chatter.settings file.

```
<?xml version="1.0" encoding="UTF-8"?>
 <ChatterSettings xmlns="http://soap.sforce.com/2006/04/metadata">
```

```
     <allowChatterGroupArchiving>true</allowChatterGroupArchiving>
     <allowRecordsInChatterGroup>true</allowRecordsInChatterGroup>
     <enableApprovalRequest>true</enableApprovalRequest>
     <enableChatter>true</enableChatter>
     <enableChatterEmoticons>true</enableChatterEmoticons>
     <enableFeedEdit>true</enableFeedEdit>
     <enableFeedsDraftPosts>false</enableFeedsDraftPosts>
     <enableFeedsRichText>true</enableFeedsRichText>
     <enableInviteCsnUsers>true</enableInviteCsnUsers>
     <enableOutOfOfficeEnabledPref>false</enableOutOfOfficeEnabledPref>
     <enableRichLinkPreviewsInFeed>true</enableRichLinkPreviewsInFeed>
     <enableTodayRecsInFeed>true</enableTodayRecsInFeed>
     <unlistedGroupsEnabled>true</unlistedGroupsEnabled>
 </ChatterSettings>
```

The following is an example `package.xml` manifest that references the ChatterSettings definitions:

```
<?xml version="1.0" encoding="UTF-8"?>
<Package xmlns="http://soap.sforce.com/2006/04/metadata">
    <types>
        <members>Chatter</members>
        <name>Settings</name>
    </types>
    <version>47.0</version>
</Package>
```

Wildcard Support in the Manifest File

The wildcard character `*` (asterisk) in the `package.xml` manifest file doesn't apply to metadata types for feature settings. The wildcard applies only when retrieving all settings, not for an individual setting. For details, see Settings. For information about using the manifest file, see Deploying and Retrieving Metadata with the Zip File.

CommunitiesSettings

Represents the feed moderation settings for an org. This type extends the Metadata metadata type and inherits its `fullName` field.

In the package manifest, all organization settings metadata types are accessed using the "Settings" name. See Settings for more details.

File Suffix and Directory Location

`CommunitiesSettings` values are stored in the `Communities.settings` file in the `settings` folder. The `.settings` files are different from other named components, because there is only one settings file for each settings component.

Version

CommunitiesSettings components are available in API version 47.0 and later.

Fields

Field Name	Field Type	Description
`canModerateAllFeedPosts`	boolean	When `true`, allows moderation features, such as flags and rules, to be set on all feed posts including those visible in communities. When set to `false`, only feed posts in communities can be moderated. Default is `false`.
`canModerateInternalFeedPosts`	boolean	When `true`, allows moderation features, such as flags and rules, to be set on record feed posts created by internal users, which may also be visible in multiple communities. Default is `false`.
`embeddedVisualforcePages`	boolean	When `true`, allows links to Visualforce pages from other Visualforce pages in the Salesforce app via the API. Default is `false`. Available in API version 48.0 and later.
`enableCommunityWorkspaces`	boolean	When `true`, allows admins to enable Experience Workspaces. Available in API version 48.0 and later.
`enableCspContactVisibilityPref`	boolean	When `true`, allows users to see contacts from private accounts that they have read access to, when the contact is controlled by the parent record. Available in API version 48.0 and later.
`enableCspNotesOnAccConPref`	boolean	When `true`, allows customer users to access notes and attachments associated with accounts and contacts. Available in API version 48.0 and later.
`enableEnablePRM`	boolean	When `true`, allows admins to enable partner users. Available in API version 48.0 and later.
`enableExternalAccHierPref`	boolean	When `true`, enables the External Account Hierarchy object. Available in API version 48.0 and later.
`enableGuestRecordReassignOrgPref`	boolean	When `true`, allows admins to set a default owner for records created by guest users. Available in API version 48.0 and later.
`enableInviteChatterGuestEnabled`	boolean	When `true`, allows guest users to be invited to use Chatter. Depricated in API version 48.0. (Deprecated)
`enableNetPortalUserReportOpts`	boolean	When `true`, allows external users in communities, with permission, to run reports. Available in API version 48.0 and later.
`enableNetworksEnabled`	boolean	When `true`, allows users to enable communities. Available in API version 47.0 and later.
`enableOotbProfExtUserOpsEnable`	boolean	When `true`, allows use of standard external profiles for self-registration and user creation. Available in API version 48.0 and later.
`enablePowerCustomerCaseStatus`	boolean	When `true`, allows users with Customer Community Plus licenses to change case status. Available in API version 48.0 and later.
`enablePRMAccRelPref`	boolean	When `true`, enables Account Relationship object and Account Relationship Data Sharing Rule setup options. Available in API version 48.0 and later.

Field Name	Field Type	Description
enableRelaxPartnerAccountFieldPref	boolean	When `true`, allows editing for Partner Account fields on and opportunities and leads. Available in API version 48.0 and later.
enableUnsupportedBrowserModalPref	boolean	When `true`, warnings are displayed for unsupported browsers in Communities. Available in API version 48.0 and later.
enableUsernameUniqForOrgPref	boolean	When `true`, username uniqueness is set at the org level. Available in API version 48.0 and later.

Declarative Metadata Sample Definition

The following is an example of a CommunitiesSettings component.

```
<CommunitiesSettings xmlns="http://soap.sforce.com/2006/04/metadata">
    <canModerateAllFeedPosts>true</canModerateAllFeedPosts>
    <canModerateInternalFeedPosts>true</canModerateInternalFeedPosts>
</CommunitiesSettings>
```

The following is an example `package.xml` that references the previous definition.

```
<Package xmlns="http://soap.sforce.com/2006/04/metadata">
<types>
<members>Communities</members>
<name>Settings</name>
</types>
<version>47.0</version>
</Package>
```

Wildcard Support in the Manifest File

The wildcard character `*` (asterisk) in the `package.xml` manifest file doesn't apply to metadata types for feature settings. The wildcard applies only when retrieving all settings, not for an individual setting. For details, see Settings. For information about using the manifest file, see Deploying and Retrieving Metadata with the Zip File.

CompanySettings

Represents global settings that affect multiple features in your organization. This type extends the Metadata metadata type and inherits its `fullName` field.

In the package manifest, all organization settings metadata types are accessed using the "Settings" name. See Settings for more details.

Declarative Metadata File Suffix and Directory Location

CompanySettings values are stored in a single file named `Company.settings` in the `settings` directory of the corresponding package directory. The `.settings` files are different from other named components because there is only one settings file for each settings component.

Version

Company Profile Settings are available in API version 27.0 and later.

Fields

Field Name	Field Type	Description
`enableCustomFiscalYear`	boolean	If a custom fiscal period is set up, this field is used to determine whether the custom fiscal period is used for forecasts. If `true`, the custom fiscal period is used. If `false` (default), standard periods are used. Available in API version 47.0 and later.
`fiscalYear`	FiscalYearSetting	The organization's fiscal year setting based on year and start month. Not available if Custom Fiscal Year or Forecasts (Classic) is enabled. When changing fiscal year settings, quotas and adjustments can be purged. For example changing your start month results in purging this data.

FiscalYearSetting

Represents your organization's fiscal year setting.

Field	Field Type	Description
`fiscalYearNameBasedOn`	string	This field is used to determine the fiscal year name. Valid values are `endingMonth` or `startingMonth`. For example, if your fiscal year starts in April 2012 and ends in March 2013, and this value is: • `endingMonth`, then 2013 is used for the fiscal year name. • `startingMonth`, then 2012 is used for the fiscal year name.
`startMonth`	string	The month on which the fiscal year is based.

Declarative Metadata Sample Definition — Fiscal Year Setting

A sample XML definition of a fiscal year setting is shown below. Note that this example is supported in API version 27.0 and later.

```xml
<?xml version="1.0" encoding="UTF-8"?>
<CompanySettings xmlns="http://soap.sforce.com/2006/04/metadata">
    <fiscalYear>
        <fiscalYearNameBasedOn>endingMonth</fiscalYearNameBasedOn>
        <startMonth>January</startMonth>
    </fiscalYear>
</CompanySettings>
```

Wildcard Support in the Manifest File

The wildcard character `*` (asterisk) in the `package.xml` manifest file doesn't apply to metadata types for feature settings. The wildcard applies only when retrieving all settings, not for an individual setting. For details, see Settings. For information about using the manifest file, see Deploying and Retrieving Metadata with the Zip File.

ConnectedAppSettings

Represents settings for connected apps. This type extends the Metadata metadata type and inherits its `fullName` field.

File Suffix and Directory Location

ConnectedAppSettings values are stored in a single file named `ConnectedApp.settings` in the `settings` directory. The `.settings` files are different from other named components because there is only one settings file for each settings component.

Version

ConnectedAppSettings components are available in API version 47.0 and later.

Fields

Field Name	Field Type	Description
`enableAdminApprovedAppsOnly`	boolean	If `false` (default), any connected app can call the Salesforce API. If `true`, only apps that have been approved or installed by the admin can call the Salesforce API.
`enableAdminApprovedAppsOnlyForExternalUser`	boolean	If `false` (default), external users can use any unblocked connected app to access the Salesforce API. If `true`, external users can't access the Salesforce API unless they use a connected app that is installed in the org and unblocked. Install and unblock connected apps on the Connected Apps OAuth Usage page.
`enableSkipUserProvisioningWizardWelcomePage`	boolean	If `false` (default), the User Provisioning Wizard Welcome page shows up when you access the wizard. To skip the welcome page in the future, you can select **Do not show me this next time**. If `true`, the Welcome page doesn't show up the next time that you access the wizard.

Declarative Metadata Sample Definition

The following is an example of a ConnectedAppSettings component.

```
<?xml version="1.0" encoding="UTF-8"?>
<ConnectedAppSettings xmlns="http://soap.sforce.com/2006/04/metadata">
    <enableAdminApprovedAppsOnly>false</enableAdminApprovedAppsOnly>

<enableAdminApprovedAppsOnlyForExternalUser>false</enableAdminApprovedAppsOnlyForExternalUser>
```

```
<enableSkipUserProvisioningWizardWelcomePage>true</enableSkipUserProvisioningWizardWelcomePage>
</ConnectedAppSettings>
```

The following is an example `package.xml` manifest that references the previous definition.

```
<?xml version="1.0" encoding="UTF-8"?>
<Package xmlns="http://soap.sforce.com/2006/04/metadata">
    <types>
        <members>ConnectedApps</members>
        <name>Settings</name>
    </types>
    <version>47.0</version>
</Package>
```

ContentSettings

Represents content settings for an org. This type extends the Metadata metadata type and inherits its `fullName` field.

In the package manifest, all organization settings metadata types are accessed using the "Settings" name. See Settings for more details.

File Suffix and Directory Location

`ContentSettings` values are stored in the `contentsettings.settings` file in the `settings` folder. The `.settings` files are different from other named components, because there is only one settings file for each settings component.

Version

ContentSettings components are available in API version 48.0 and later.

Fields

Field Name	Field Type	Description
`enableChatterFileLink`	boolean	When `true`, allows users to share files via links. When set to `false`, users can't use file sharing. Default is `true`.
`enableCMSC2CConnections`	boolean	When `true`, allows org to enable Salesforce CMS Cloud to Cloud Connections.
`enableContent`	boolean	When `true`, allows org to enable Content.
`enableContentAutoAssign`	boolean	When `true`, allows org to auto assign Content feature licenses to users.
`enableContentDistForPortalUsers`	boolean	When `true`, allows portal users to create Content Deliveries for managed files in a library.
`enableContentDistPwOptionsBit1`	boolean	When `true`, allow using with `ContentDistPasswordOptionsBit2` to set up one of three possible delivery security options.
`enableContentDistPwOptionsBit2`	boolean	When `true`, allow using with `ContentDistPasswordOptionsBit1` to set up one of three possible delivery security options.

Field Name	Field Type	Description
`enableContentDistribution`	boolean	When `true`, allows the Content Delivery user permission to be enabled for users. Content deliveries let users create links to share files externally, with optional security settings.
`enableContentSupportMultiLanguage`	boolean	When `true`, enables content to support multiple languages.
`enableContentWorkspaceAccess`	boolean	When `true`, content libraries are visible in the API and UI for users who have read access to libraries, even if they don't have access to the original Salesforce CRM Content app.
`enableDeleteFileInContentPacks`	boolean	When `true`, allows a file's owner to delete the file that is included in one or more content packs.
`enableFileShareSetByRecord`	boolean	When `true`, files shared to records will default to Set by Record.
`enableFilesUsrShareNetRestricted`	boolean	When `true`, files respect user sharing settings. Files shared with users with **SharedUsers** visibility are only accessible to users who are members of the community the file was created in.
`enableJPGPreviews`	boolean	When `true`, attempts to use other SVG alternative formats such as JPG as preview images.
`enableLibraryManagedFiles`	boolean	When `true`, controls the ability to publish files created in Chatter with a Content Library (ContentWorkspace). The library can manage the file.
`enableShowChatterFilesInContent`	boolean	When `true`, allows users to search for Chatter files in content.
`enableSiteGuestUserToUploadFiles`	boolean	When `true`, site guest users can upload files.
`enableUploadFilesOnAttachments`	boolean	When `true`, shows the **New File** button on the Attachments related lists to upload files, rather than legacy Attachments.
`skipContentAssetTriggers`	boolean	When `true`, disables content trigger execution and custom validation on content assets.
`skipContentAssetTriggersOnDeploy`	boolean	When `true`, disables content trigger execution when deploying content assets.

Declarative Metadata Sample Definition

This is a sample content settings file.

```
<?xml version="1.0" encoding="UTF-8"?>
<ContentSettings xmlns=</ContentSettings>
```

Wildcard Support in the Manifest File

The wildcard character `*` (asterisk) in the `package.xml` manifest file doesn’t apply to metadata types for feature settings. The wildcard applies only when retrieving all settings, not for an individual setting. For details, see Settings. For information about using the manifest file, see Deploying and Retrieving Metadata with the Zip File.

ContractSettings

Represents contract settings. For more information, see “Set Up Contracts” in the Salesforce online help.

In the package manifest, all organization settings metadata types are accessed using the “Settings” name. See Settings for more details.

File Suffix and Directory Location

There is one contract settings file stored in a file named `Contract.settings` in the `settings` directory. The `.settings` files are different from other named components because there is only one settings file for each settings component.

Version

ContractSettings is available in API version 27.0 and later.

Fields

Field Name	Field Type	Description
`autoCalculateEndDate`	boolean	Indicates whether the end date of a contract is automatically calculated (`true`) or not (`false`).
`notifyOwnersOnContractExpiration`	boolean	Indicates whether account and contract owners are automatically sent email notifications when a contract expires (`true`) or not (`false`).

Declarative Metadata Sample Definition

This is a sample contract settings file.

```
<?xml version="1.0" encoding="UTF-8"?>
<ContractSettings xmlns="http://soap.sforce.com/2006/04/metadata">
    <autoCalculateEndDate>true</autoCalculateEndDate>
    <notifyOwnersOnContractExpiration>false</notifyOwnersOnContractExpiration>
</ContractSettings>
```

Wildcard Support in the Manifest File

The wildcard character `*` (asterisk) in the `package.xml` manifest file doesn’t apply to metadata types for feature settings. The wildcard applies only when retrieving all settings, not for an individual setting. For details, see Settings. For information about using the manifest file, see Deploying and Retrieving Metadata with the Zip File.

CurrencySettings

Represents an organization’s currency settings, including supporting multiple currencies and currency effective dates. This type extends the Metadata metadata type and inherits its `fullName` field.

In the package manifest, all organization settings metadata types are accessed using the “Settings” name. See Settings for more details.

File Suffix and Directory Location

CurrencySettings values are stored in the `Currency.settings` file in the `settings` directory. The `.settings` files are different from other named components because there is only one settings file for each settings component.

Version

CurrencySettings is available in API version 47.0 and later.

Fields

Field Name	Field Type	Description
`enableCurrencyEffectiveDates`	boolean	Indicates whether effective dated currency is enabled (`true`) or not (`false`). This field has a default value of `false`. To enable this preference, `enableMultiCurrency` must be set to `true`
`enableCurrencySymbolWithMultiCurrency`	boolean	Indicates whether the currency symbol (`true`) or ISO code (`false`) displays in multi-currency orgs. This field has a default value of `false`. This field has no effect if `enableMultiCurrency` is set to `false`.
`enableMultiCurrency`	boolean	Indicates whether multiple currencies are enabled (`true`) or not (`false`). This field has a default value of `false`. Note: Once set to `true`, this field can't be set to `false`. See Considerations for Enabling Multiple Currencies for more information.
`isMultiCurrencyActivationAllowed`	boolean	Indicates whether Salesforce Support must activate multiple currencies (`true`) or a system administrator can activate it (`false`). This field has a default value of `false`. Setting this field to `true` provides an extra layer of protection against accidentally enabling multiple currencies. Note: If Customizable Forecasting is enabled, this field is set to `false`, and you must contact Salesforce Support to activate multiple currencies.
`isParenCurrencyConvDisabled`	boolean	Indicates whether parenthetical currency conversion is disabled (`true`) or enabled (`false`). This field has a default value of `true`. When this field is set to `false`, Salesforce displays converted currency amounts to users whose personal currency differs from the currency of the record they're viewing.

Declarative Metadata Sample Definition

The following is an example of a CurrencySettings file.

```
<?xml version="1.0" encoding="UTF-8"?>
<CurrencySettings xmlns="http://soap.sforce.com/2006/04/metadata">
    <enableCurrencyEffectiveDates>false</enableCurrencyEffectiveDates>
    <enableCurrencySymbolWithMultiCurrency>false</enableCurrencySymbolWithMultiCurrency>
    <enableMultiCurrency>false</enableMultiCurrency>
    <isMultiCurrencyActivationAllowed>false</isMultiCurrencyActivationAllowed>
    <isParenCurrencyConvDisabled>false</isParenCurrencyConvDisabled>
</CurrencySettings>
```

The following is an example `package.xml` that references the previous definition.

```
<?xml version="1.0" encoding="UTF-8"?>
<Package xmlns="http://soap.sforce.com/2006/04/metadata">
    <types>
        <members>Currency</members>
        <name>Settings</name>
    </types>
    <version>47.0</version>
</Package>
```

Wildcard Support in the Manifest File

The wildcard character `*` (asterisk) in the `package.xml` manifest file doesn't apply to metadata types for feature settings. The wildcard applies only when retrieving all settings, not for an individual setting. For details, see Settings. For information about using the manifest file, see Deploying and Retrieving Metadata with the Zip File.

DataDotComSettings

Represents the org's Data.com settings. This type extends the Metadata metadata type and inherits its `fullName` field.

File Suffix and Directory Location

DataDotComSettings values are stored in the `DataDotCom.settings` file in the `settings` folder.

Version

DataDotComSettings components are available in API version 47.0 and later.

Fields

Field Name	Field Type	Description
`enableAccountExportButtonOff`	boolean	Indicates whether Account Export to Excel is enabled for Prospector (`true`) or not (`false`). Default value is `false`.
`enableAccountImportButtonOff`	boolean	Indicates whether Account Import to CRM is enabled for Prospector (`true`) or not (`false`). Default value is `false`.
`enableAllowDupeContactFromLead`	boolean	Indicates whether Prospector Lead Import Duplicate Check is enabled (`true`) or not (`false`). Default value is `false`.
`enableAllowDupeLeadFromContact`	boolean	Indicates whether Prospector Contact Import Duplicate Check is enabled (`true`) or not (`false`). Default value is `false`.
`enableCleanUpgradeRequested`	boolean	This field is no longer in use.

Field Name	Field Type	Description
enableContactExportButtonOff	boolean	Indicates whether Contact Export to Excel is enabled for Prospector (true) or not (false). Default value is false.
enableContactImportButtonOff	boolean	Indicates whether Contact Import to CRM is enabled for Prospector (true) or not (false). Default value is false.

Declarative Metadata Sample Definition

The following is an example of a DataDotComSettings component.

```xml
<?xml version="1.0" encoding="UTF-8"?>
<DataDotComSettings xmlns="http://soap.sforce.com/2006/04/metadata">
 <enableAccountExportButtonOff>true</enableAccountExportButtonOff>
 <enableAccountImportButtonOff>true</enableAccountImportButtonOff>
 <enableAllowDupeContactFromLead>true</enableAllowDupeContactFromLead>
 <enableAllowDupeLeadFromContact>true</enableAllowDupeLeadFromContact>
 <enableContactExportButtonOff>true</enableContactExportButtonOff>
 <enableContactImportButtonOff>true</enableContactImportButtonOff>
</DataDotComSettings>
```

The following is an example `package.xml` that references the previous definition.

```xml
<?xml version="1.0" encoding="UTF-8"?>
<Package xmlns="http://soap.sforce.com/2006/04/metadata">
 <types>
  <members>DataDotCom</members>
  <name>Settings</name>
 </types>
 <version>29.0</version>
</Package>
```

DeploymentSettings

Represents the settings affecting how deployments behave in the org. This type extends the Metadata metadata type and inherits its `fullName` field.

In the package manifest, all organization settings metadata types are accessed using the “Settings” name. See Settings for more details.

File Suffix and Directory Location

DeploymentSettings values are stored in the `Deployment.settings` file in the `settings` directory. The `.settings` files are different from other named components, because there is only one settings file for each settings component.

Version

DeploymentSettings components are available in API version 47.0 and later.

Fields

Field Name	Field Type	Description
doesSkipAsyncApexValidation	boolean	Indicates whether deployments from this org skip asynchronous Apex validations (`true`) or not (`false`). The default value is `true`. Set this field to `false` when an Apex class in the package you're deploying is used by an Apex batch job that could run during the deployment. The deployment of a package containing an Apex class that is used by a running batch job fails without validation.

Declarative Metadata Sample Definition

The following is an example of a DeploymentSettings component.

```
<?xml version="1.0" encoding="UTF-8"?>
<DeploymentSettings xmlns="http://soap.sforce.com/2006/04/metadata">
   <doesSkipAsyncApexValidation>true</doesSkipAsyncApexValidation>
</DeploymentSettings>
```

The following is an example `package.xml` that references the previous definition.

```
<?xml version="1.0" encoding="UTF-8"?>
<Package xmlns="http://soap.sforce.com/2006/04/metadata">
   <types>
      <members>Deployment</members>
      <name>Settings</name>
   </types>
   <version>47.0</version>
</Package>
```

Wildcard Support in the Manifest File

The wildcard character `*` (asterisk) in the `package.xml` manifest file doesn't apply to metadata types for feature settings. The wildcard applies only when retrieving all settings, not for an individual setting. For details, see Settings. For information about using the manifest file, see Deploying and Retrieving Metadata with the Zip File.

DevHubSettings

Represents Developer Hub (Dev Hub) settings. This type extends the Metadata metadata type and inherits its `fullName` field.

In the package manifest, all organization settings metadata types are accessed using the "Settings" name. See Settings for more details.

File Suffix and Directory Location

DevHubSettings values are stored in the `DevHub.settings` file in the `settings` folder. The `.settings` files are different from other named components because there is only one settings file for each settings component.

Version

DevHubSettings are available in API version 47.0 and later.

Fields

Field Name	Field Type	Description
`enablePackaging2`	boolean	Indicates whether unlocked and second-generation managed packaging is enabled `true` or not `false`. To enable `enablePackaging2`, first enable `enableScratchOrgManagementPref`.
`enableScratchOrgManagementPref`	boolean	Indicates whether Dev Hub is enabled `true` or not `false`. When enabled, a Dev Hub allows you to create and manage scratch orgs.

Declarative Metadata Sample Definition

The following is an example of a DevHubSettings component.

```
<?xml version="1.0" encoding="UTF-8"?>
<DevHubSettings xmlns="http://soap.sforce.com/2006/04/metadata">
    <enablePackaging2>true</enablePackaging2>
    <ScratchOrgManagementPref>true</ScratchOrgManagementPref>
</DevHubSettings>
```

Wildcard Support in the Manifest File

The wildcard character `*` (asterisk) in the `package.xml` manifest file doesn't apply to metadata types for feature settings. The wildcard applies only when retrieving all settings, not for an individual setting. For details, see Settings. For information about using the manifest file, see Deploying and Retrieving Metadata with the Zip File.

EACSettings

Represents the Einstein Activity Capture metadata type. Use Einstein Activity Capture to add emails and events from your Microsoft or Google account to the activity timeline of related Salesforce records. Automatically sync contact and event data between your Microsoft or Google account and Salesforce. This type extends the Metadata metadata type and inherits its `fullName` field.

In the package manifest, all organization settings metadata types are accessed using the "Settings" name. See Settings for more details.

File Suffix and Directory Location

`EACSettings` components have the suffix `EAC` and are stored in the `Settings` folder.

Version

`EACSettings` components are available in API version 48.0 and later.

Fields

Field Name	Field Type	Description
`enableActivityCapture`	boolean	Indicates whether Einstein Activity Capture is enabled (`true`) or not (`false`). `provisionProductivityFeatures` must be `true` to use this feature. To ensure that your org's requirements for handling sensitive data are met, read "Considerations for Setting Up Einstein Activity Capture" in Salesforce Help. Default value is `false`.
`enableActivityMetrics`	boolean	Indicates whether Activity Metrics are enabled (`true`) or not (`false`). `enableActivityCapture` must be `true` to use this feature. Before enabling this feature, see "Considerations for Using Activity Metrics" in Salesforce Help. Default value is `false`.
`enableActivitySyncEngine`	boolean	Indicates whether combined sync and capture is enabled for events, contacts, and emails (`true`) or not (`false`). Default value is `false`.
`enableEACForEveryonePref`	boolean	Indicates whether users who have the `enableActivityCapture` set to `false` can still see emails and events in their Salesforce timeline (`true`) or not (`false`). Default value is `true`.
`enableInboxActivitySharing`	boolean	Indicates whether the default activity sharing for new users is set to `Everyone` (`true`) or not (`false`). For example, if `enableInboxActivitySharing` is `true`, then new Einstein Activity Capture users have their activity sharing set to `Everyone` by default. This setting does not affect the activity sharing setting of existing users. Default value is `true`.
`enableInsightsInTimeline`	boolean	Indicates whether Email Insights is enabled (`true`) or not (`false`). Default value is `true`.
`enableInsightsInTimelineEacStd`	boolean	Indicates whether Email Insights is enabled for users with an Einstein Activity Capture Standard permission set (`true`) or not (`false`). For more information, see "Turn On Einstein Email Insights" in Salesforce Help. Default value is `false`.
`provisionProductivityFeatures`	boolean	Indicates whether your org is ready for productivity features to be enabled (`true`) or not (`false`). Default value is `false`.

Declarative Metadata Sample Definition

The following is an example of the EAC.settings file:

```
<?xml version="1.0" encoding="UTF-8"?>
<EACSettings xmlns="http://soap.sforce.com/2006/04/metadata">
 <enableActivityCapture>true</enableActivityCapture>
 <enableActivityMetrics>true</enableActivityMetrics>
 <enableActivitySyncEngine>true</enableActivitySyncEngine>
 <enableEACForEveryonePref>true</enableEACForEveryonePref>
 <enableInboxActivitySharing>true</enableInboxActivitySharing>
 <enableInsightsInTimeline>true</enableInsightsInTimeline>
 <enableInsightsInTimelineEacStd>true</enableInsightsInTimelineEacStd>
 <provisionProductivityFeatures>true</provisionProductivityFeatures>
</EACSettings>
```

Example Package Manifest

The following is an example package manifest used to deploy or retrieve the EAC settings metadata:

```
<?xml version="1.0" encoding="UTF-8"?>
<Package xmlns="http://soap.sforce.com/2006/04/metadata">
 <types>
  <members>EAC</members>
  <name>Settings</name>
 </types>
 <version>29.0</version>
</Package>
```

Wildcard Support in the Manifest File

The wildcard character `*` (asterisk) in the `package.xml` manifest file doesn't apply to metadata types for feature settings. The wildcard applies only when retrieving all settings, not for an individual setting. For details, see Settings. For information about using the manifest file, see Deploying and Retrieving Metadata with the Zip File.

EmailAdministrationSettings

Represents an organization's email administration settings, including email deliverability, security compliance, relay configurations, and system notifications. This type extends the Metadata metadata type and inherits its `fullName` field.

In the package manifest, all organization settings metadata types are accessed using the "Settings" name. See Settings for more details.

File Suffix and Directory Location

EmailAdministrationSettings values are stored in the `EmailAdminstration.settings` file in the `settings` directory. The `.settings` files are different from other named components because there is only one settings file for each settings component.

Version

EmailAdministrationSettings is available in API version 47.0 and later.

Fields

Field Name	Field Type	Description
`enableComplianceBcc`	boolean	Indicates whether a copy of each outbound email message is sent to an email address you specify (`true`) or not (`false`). This field has a default value of `false`. Note: To use this feature, you must specify an email address in **Compliance BCC Email** in Setup.
`enableEmailConsentManagement`	boolean	Indicates whether Enforce Email Privacy Settings is enabled `(true)` or not `(false)`. When enabled, Salesforce respects each recipient's email privacy preferences. Default value is `false`.
`enableEmailSenderIdCompliance`	boolean	Indicates whether outgoing emails comply with Sender ID email protocols (`true`) or not (`false`). This field has a default value of `false`. To enable this preference, `enableEmailSpfCompliance` must be set to `true`. Note: Evaluate the multiple standard email security protocols (SPF, DKIM, and DMARC) supported by Salesforce before you enable this setting.
`enableEmailSpfCompliance`	boolean	Indicates whether outgoing emails comply with Sender Policy Framework (SPF) email authentication (`true`) or not (`false`). This field has a default value of `true`.
`enableEmailToSalesforce`	boolean	Indicates whether Email to Salesforce is enabled (`true`) or disabled (`false`). This field has a default value of `false`.
`enableEmailWorkflowApproval`	boolean	Indicates whether users can respond to email approval requests directly from their email (`true`) or not (`false`). This field has a default value of `false`. See Considerations for Email Approval before enabling this field.
`enableEnhancedEmailEnabled`	boolean	Indicates whether Enhanced Email is enabled `(true)` or not `(false)`. Default value is `true`.
`enableHandleBouncedEmails`	boolean	Indicates whether emails sent from Salesforce to an invalid email address bounce back to Salesforce (`true`) or not (`false`). This field has a default value of `true`. With bounce handling enabled, reps know which lead, contact, or person account has a bad email address, and they know which specific email wasn't delivered.
`enableHtmlEmail`	boolean	Indicates whether users receive Email-To-Case emails in HTML format (`true`) or receive a text version instead (`false`). This field has a default value of `false`. When this field is set to `true`, users receive a warning message

Field Name	Field Type	Description
		about potential malicious HTML before they view incoming HTML email content.
enableListEmailLogActivities	boolean	Indicates whether Salesforce logs sent list emails as activities (true) or not (false). Default value is true.
enableResendBouncedEmails	boolean	Indicates whether the system forwards a copy of each bounced email message to the sender (true) or only displays the bounce alert (false). This field has a default value of false. To enable this preference, enableHandleBouncedEmails must be set to true.
enableRestrictTlsToDomains	boolean	Indicates whether the selected Transport Layer Security (TLS) setting applies only to specific domains (true) or applies to all domains (false). This field has a default value of false. Note: To enable this preference, you must specify a **TLS Setting** other than Preferred and provide the comma-separated list of domains through **Deliverability** in Setup. When this field is set to true, any domains not in the list use the system default TLS Setting of Preferred.
enableSendThroughGmailPref	boolean	Deprecated.
enableSendViaExchangePref	boolean	Indicates whether users can use Office 365 to send emails (true) or not (false). Default value is false.
enableSendViaGmailPref	boolean	Indicates whether users can use Gmail to send emails (true) or not (false). Default value is false.
enableSetMatchingEmailsOnBounce	boolean	Indicates whether a bounce alert is displayed next to all instances of the email address (true) or only on the record from which the email was originally sent (false). This field has a default value of false. To enable this preference, enableHandleBouncedEmails must be set to true.
enableUseOrgFootersForExtTrans	boolean	Indicates whether emails sent through external email services (such as Gmail or Office 365) include the Salesforce footer (true) or not (false). This field has a default value of false.
sendEmailsEvenWhenAutomationUpdatesSameRecord	boolean	Indicates whether to prevent automated field updates from suppressing email notifications. When the value is true, email notifications are sent even when a record creation or update triggers automated updates to the same record. When the value is false, email notifications aren't sent

Field Name	Field Type	Description
		when a record creation or update triggers a process, workflow rule, or Apex trigger to update the same record. Corresponds to the Stop Automated Field Updates from Suppressing Email Notifications critical update.
`sendMassEmailNotification`	boolean	Indicates whether users receive an auto-generated status email from Salesforce for each mass email they send (`true`) or not (`false`). This field has a default value of `true`.
`sendTextOnlySystemEmails`	boolean	Indicates whether all system emails are sent via text only (`true`) or allow other formats (`false`). This field has a default value of `false`.

Declarative Metadata Sample Definition

The following is an example of an EmailAdministrationSettings component.

```xml
<?xml version="1.0" encoding="UTF-8"?>
<EmailAdministrationSettings xmlns="http://soap.sforce.com/2006/04/metadata">
    <enableEmailWorkflowApproval>false</enableEmailWorkflowApproval>
    <enableComplianceBcc>false</enableComplianceBcc>
    <enableEmailSenderIdCompliance>false</enableEmailSenderIdCompliance>
    <enableEmailSpfCompliance>true</enableEmailSpfCompliance>
    <enableEmailToSalesforce>false</enableEmailToSalesforce>
    <enableHandleBouncedEmails>true</enableHandleBouncedEmails>
    <enableHtmlEmail>true</enableHtmlEmail>
    <enableResendBouncedEmails>false</enableResendBouncedEmails>
    <enableRestrictTlsToDomains>false</enableRestrictTlsToDomains>
    <sendMassEmailNotification>true</sendMassEmailNotification>
    <enableSetMatchingEmailsOnBounce>false</enableSetMatchingEmailsOnBounce>
    <sendTextOnlySystemEmails>false</sendTextOnlySystemEmails>
    <enableUseOrgFootersForExtTrans>false</enableUseOrgFootersForExtTrans>
    <enableSendViaGmailPref>false</enableSendViaGmailPref>
    <enableSendViaExchangePref>true</enableSendViaExchangePref>
    <enableListEmailLogActivities>false</enableListEmailLogActivities>
    <enableEnhancedEmailEnabled>false</enableEnhancedEmailEnabled>
    <enableEmailConsentManagement>false</enableEmailConsentManagement>
</EmailAdministrationSettings>
```

The following is an example `package.xml` that references the previous definition.

```xml
<?xml version="1.0" encoding="UTF-8"?>
<Package xmlns="http://soap.sforce.com/2006/04/metadata">
    <types>
        <members>EmailAdministration</members>
        <name>Settings</name>
    </types>
    <version>47.0</version>
</Package>
```

Wildcard Support in the Manifest File

The wildcard character `*` (asterisk) in the `package.xml` manifest file doesn't apply to metadata types for feature settings. The wildcard applies only when retrieving all settings, not for an individual setting. For details, see Settings. For information about using the manifest file, see Deploying and Retrieving Metadata with the Zip File.

EmailIntegrationSettings

Represents an org's settings for the Outlook integration, Gmail integration, and Salesforce Inbox. This type extends the Metadata metadata type and inherits its `fullName` field.

In the package manifest, all organization settings metadata types are accessed using the "Settings" name. See Settings for more details.

File Suffix and Directory Location

EmailIntegrationSettings values are stored in the `EmailIntegration.settings` file in the `settings` directory. The `.settings` files are different from other named components because there is only one settings file for each settings component.

Version

EmailIntegrationSettings fields are available in API version 47.0 and later.

Fields

Field Name	Field Type	Description
`doesEmailLogAsEmailMessageInOutlook`	boolean	Indicates whether the Outlook integration logs emails to the Email Message object (`true`) or as tasks (`false`). The default value is `true`. This field can only be used if the `enableOutlookIntegration` field is set to `true`.
`doesGmailStayConnectedToSalesforce`	boolean	Indicates whether Gmail integration users log in to Salesforce from Gmail each time their sessions expires. If set to `true`, users log in from the Gmail integration once, and their credentials are remembered the next time they use the Gmail integration. If set to `false`, users log in to Salesforce from the Gmail integration each time their Salesforce session expires. The default value is `false`. This field can only be used if the `enableGmailIntegration` field is set to `true`.
`enableContactAndEventSync`	boolean	Indicates whether users can sync calendar events and contacts between Salesforce and their Microsoft and Google accounts (`true`) or not (`false`). Lightning Sync or Einstein Activity Capture must be enabled to use this feature. The default value is `false`. Available in API version 48.0 and later.

Field Name	Field Type	Description
enableEmailTrackingInMobile	boolean	Indicates whether Salesforce Inbox mobile users can track emails (true) or not (false) while in the Inbox mobile app. The default value is true. This setting applies only to email tracking in the Salesforce Inbox mobile app. To disable the feature in Outlook or Gmail, use the Lightning App Builder. Remove the Email Tracking - Compose and Email Tracking - List components from the email application pane after setting the isLayoutCustomizationAllowed field to true.
enableEngageForOutlook	boolean	Indicates whether Engage For Outlook is enabled (true) or not (false). When set to true, Engage users can connect their Outlook account and send Engage emails from their Outlook inbox.
enableGmailIntegration	boolean	Indicates whether the Gmail integration is enabled (true) or not (false). When set to true, G Suite users with the Gmail integration can connect their Gmail account and work with Salesforce data in their email. The default value is true.
enableOutlookIntegration	boolean	Indicates whether the Outlook integration is enabled (true) or not (false). When set to true, Outlook users with the Outlook integration can connect their Outlook account and work with Salesforce data in their email. The default value is false.
enableProductivityFeatures	boolean	Indicates whether Inbox features, such as Availability and Send later, are available (true) or not available (false) in the Outlook or Gmail integration. The default value is false. This field can only be used if either the enableOutlookIntegration or enableGmailIntegration field is set to true and if the org has an Inbox license. Note: To see Inbox features, users must also have either the Inbox with Einstein Activity Capture or the Inbox without Einstein Activity capture permission set.
enableSupplementalContactInfoInMobile	boolean	Indicates whether Salesforce Inbox mobile app users see third-party contact information when contacts are shown (true) or not (false) in the Inbox mobile app. The default value is false.
isLayoutCustomizationAllowed	boolean	Indicates whether Salesforce admins are allowed (true) or not allowed (false) to create custom email application panes using the Lightning App Builder. The email application pane defines the layout of the Salesforce pane in Outlook and Gmail. The default value is true. This field can only be used if either the enableOutlookIntegration or enableGmailIntegration field is set to true.

Field Name	Field Type	Description
shouldUseTrustedDomainsList	boolean	Indicates if the web domains listed in the Outlook Integration & Sync page in Salesforce Setup are supported (`true`) or not (`false`). These domains are for users who access email using Outlook on the web. The default value is `true`. This field can only be used if the `enableOutlookIntegration` field is set to `true`.

Declarative Metadata Sample Definition

The following is an example of a EmailIntegrationSettings file.

```
<?xml version="1.0" encoding="UTF-8"?>
<EmailIntegrationSettings xmlns="http://soap.sforce.com/2006/04/metadata">
        <enableContactAndEventSync>true</enableContactAndEventSync>
        <enableProductivityFeatures>true</enableProductivityFeatures>
        <doesGmailStayConnectedToSalesforce>true</doesGmailStayConnectedToSalesforce>
        <enableOutlookIntegration>true</enableOutlookIntegration>
        <enableGmailIntegration>true</enableGmailIntegration>
        <isLayoutCustomizationAllowed>true</isLayoutCustomizationAllowed>
        <doesEmailLogAsEmailMessageInOutlook>false</doesEmailLogAsEmailMessageInOutlook>
        <shouldUseTrustedDomainsList>false</shouldUseTrustedDomainsList>
        <enableEmailTrackingInMobile>true</enableEmailTrackingInMobile>
       <enableSupplementalContactInfoInMobile>false</enableSupplementalContactInfoInMobile>

        <enableEngageForOutlook>true</enableEngageForOutlook>
</EmailIntegrationSettings>
```

The following is an example `package.xml` that references the previous definition.

```
<?xml version="1.0" encoding="UTF-8"?>
<Package xmlns="http://soap.sforce.com/2006/04/metadata">
    <types>
        <members>EmailIntegration</members>
        <name>Settings</name>
    </types>
    <version>47.0</version>
</Package>
```

Wildcard Support in the Manifest File

The wildcard character `*` (asterisk) in the `package.xml` manifest file doesn't apply to metadata types for feature settings. The wildcard applies only when retrieving all settings, not for an individual setting. For details, see Settings. For information about using the manifest file, see Deploying and Retrieving Metadata with the Zip File.

EmailTemplateSettings

Represents an org's email template settings. This type extends the Metadata metadata type and inherits its `fullName` field.

In the package manifest, all organization settings metadata types are accessed using the "Settings" name. See Settings for more details.

File Suffix and Directory Location

EmailTemplateSettings values are stored in the `EmailTemplate.settings` file in the `settings` directory. The `.settings` files are different from other named components because there is only one settings file for each settings component.

Version

EmailTemplateSettings is available in API version 47.0 and later.

Fields

Field Name	Field Type	Description
`enableTemplateEnhancedFolderPref`	boolean	Indicates whether Folders and Enhanced Sharing for Email Templates is enabled (`true`) or not (`false`). This feature allows users to create and manage folders for email templates. Default value is `false`.

Declarative Metadata Sample Definition

The following is an example of the package file.

```
<?xml version="1.0" encoding="UTF-8"?>
<Package xmlns="http://soap.sforce.com/2006/04/metadata">
    <types>
        <members>EmailTemplate</members>
        <name>Settings</name>
    </types>
</Package>
```

The package file references the following EmaillTemplate.settings file.

```
<?xml version="1.0" encoding="UTF-8"?>
<EmailTemplateSettings xmlns="http://soap.sforce.com/2006/04/metadata">
    <enableTemplateEnhancedFolderPref>true</enableTemplateEnhancedFolderPref>
</EmailTemplateSettings>
```

Wildcard Support in the Manifest File

The wildcard character `*` (asterisk) in the `package.xml` manifest file doesn't apply to metadata types for feature settings. The wildcard applies only when retrieving all settings, not for an individual setting. For details, see Settings. For information about using the manifest file, see Deploying and Retrieving Metadata with the Zip File.

EnhancedNotesSettings

Represents an org's enhanced note settings, such as enabling enhanced notes and enabling tasks in enhanced notes.This type extends the Metadata metadata type and inherits its `fullName` field.

File Suffix and Directory Location

EnhancedNotesSettings values are stored in the `EnhancedNotes.settings` file in the `settings` directory.

In the package manifest, all organization settings metadata types are accessed using the “Settings” name. See Settings for more details.

Version

EnhancedNotesSettings is available in API version 47.0 and later.

Fields

Field Name	Field Type	Description
`enableEnhancedNotes`	boolean	Indicates whether enhanced notes are enabled `(true)` or not `(false)`. With enhanced notes, users can relate a note to multiple records, access version history, and enjoy enhanced format options. Users must have the Use New Notes permission to use enhanced notes. Default value is `true`.
`enableTasksOnEnhancedNotes`	boolean	Indicates whether users can create tasks from notes `(true)` or not `(false)`. In the Salesforce app, users can create a task from a note by swiping a line on the note. Alternatively, they can tap in the toolbar to add or update the status of an action item. Users must have the Use New Notes permission to use enhanced notes. Default value is `true`.

Declarative Metadata Sample Definition

The following is an example of the `EnhancedNotesSettings.settings` file:

```
<?xml version="1.0" encoding="UTF-8"?>
<EnhancedNotesSettings xmlns="http://soap.sforce.com/2006/04/metadata">
 <enableEnhancedNotes>true</enableEnhancedNotes>
 <enableTasksOnEnhancedNotes>true</enableTasksOnEnhancedNotes>
</EnhancedNotesSettings>
```

Example Package Manifest

The following is an example package manifest used to deploy or retrieve the EnhancedNotesSettings settings metadata:

```
<?xml version="1.0" encoding="UTF-8"?>
<Package xmlns="http://soap.sforce.com/2006/04/metadata">
 <types>
  <members>EnhancedNotesSettings</members>
  <name>Settings</name>
 </types>
 <version>47.0</version>
</Package>
```

EncryptionKeySettings

Represents an org's encryption key settings, such as customer-supplied keys options and key derivation settings. This type extends the Metadata metadata type and inherits its `fullName` field.

In the package manifest, all organization settings metadata types are accessed using the "Settings" name. See Settings for more details.

File Suffix and Directory Location

EncryptionKeySettings values are stored in the `EncryptionKey.settings` file in the `settings` folder. The `.settings` files are different from other named components because there is only one settings file for each settings component.

Version

EncryptionKeySettings is available in API versions 47.0 and later.

Special Access Rules

To enable EncryptionKeySettings, you need the Customize Application and Manage Encryption Keys permissions.

Fields

Field Name	Field Type	Description
`enableCacheOnlyKeys`	boolean	Indicates whether the Cache-Only Key Service is available (`true`) or not (`false`). The default value is `false`. If set to `true`, users can configure a cache-only key callout connection and apply key material stored outside of Salesforce to data on demand.
`canOptOutOfDerivationWithBYOK`	boolean	Indicates that users can opt out of key derivation processes on a key-by-key basis when they upload key material (`true`) or can't (`false`). The default value is `false`.
`enableReplayDetection`	boolean	Indicates whether cache-only key callouts are protected from replay attacks by a nonce (`true`) or not (`false`). Requires `enableCacheOnlyKeys=”true”` before setting `enableReplayDetection` to `true`.

Declarative Metadata Sample Definition

The following is an example of the EncryptionKey.settings file:

```
<?xml version="1.0" encoding="UTF-8"?>
<EncryptionKeySettings xmlns="http://soap.sforce.com/2006/04/metadata">
    <enableCacheOnlyKeys>true</enableCacheOnlyKeys>
    <enableReplayDetection>true</enableReplayDetection>
    <canOptOutOfDerivationWithBYOK>true</canOptOutOfDerivationWithBYOK>
</EncryptionKeySettings>
```

Example Package Manifest

The following is an example package manifest used to deploy or retrieve the encryption key settings metadata for an organization:

```
<?xml version="1.0" encoding="UTF-8"?>
<Package xmlns="http://soap.sforce.com/2006/04/metadata">
    <types>
        <members>EncryptionKeys</members>
        <name>Settings</name>
    </types>
    <version>47.0</version>
</Package>
```

Wildcard Support in the Manifest File

The wildcard character * (asterisk) in the `package.xml` manifest file doesn't apply to metadata types for feature settings. The wildcard applies only when retrieving all settings, not for an individual setting. For details, see Settings. For information about using the manifest file, see Deploying and Retrieving Metadata with the Zip File.

EntitlementSettings

Represents an organization's entitlement settings.

In the package manifest, all organization settings metadata types are accessed using the "Settings" name. See Settings for more details.

File Suffix and Directory Location

EntitlementSettings values are stored in the `Entitlements.settings` file in the `settings` directory. The `.settings` files are different from other named components because there is only one settings file for each settings component.

Version

EntitlementSettings is available in API version 27.0 and later.

Fields

Field Name	Field Type	Description
`assetLookupLimitedToActiveEntitlementsOnAccount`	boolean	Indicates whether entitlements-related lookup filters on cases return only the assets related to the active entitlements on the case's account (`true`) or not (`false`).
`assetLookupLimitedToActiveEntitlementsOnContact`	boolean	Indicates whether entitlements-related lookup filters on cases return only the assets related to the active entitlements on the case's contact (`true`) or not (`false`).
`assetLookupLimitedToSameAccount`	boolean	Indicates whether entitlements-related lookup filters on cases return only the assets related to the case's account (`true`) or not (`false`).

Field Name	Field Type	Description
`assetLookupLimitedToSameContact`	boolean	Indicates whether entitlements-related lookup filters on cases return only the assets related to the case's contact (`true`) or not (`false`).
`enableEntitlements`	boolean	Indicates whether entitlements are enabled (`true`) or not (`false`).
`enableEntitlementVersioning`	boolean	Indicates whether entitlement versioning is enabled (`true`) or not (`false`). This field is available in API version 28.0 and later.
`enableMilestoneFeedItem`	boolean	When set to `true`, indicates whether to post to the feed and the record owner's profile page when a milestone is completed or violated. When set to `false`, indicates that no post occurs when a milestone is completed or violated. This field is available in API version 47.0 and later.
`enableMilestoneStoppedTime`	boolean	Indicates whether to show the **Stopped Time** and **Actual Elapsed Time** fields on an entitlement milestone (`true`) or not (`false`). This field is available in API version 47.0 and later.
`entitlementLookupLimitedToActiveStatus`	boolean	Indicates whether entitlements-related lookup filters on cases return only active entitlements (`true`) or not (`false`).
`entitlementLookupLimitedToSameAccount`	boolean	Indicates whether entitlements-related lookup filters on cases return only the entitlements related to the case's account (`true`) or not (`false`).
`entitlementLookupLimitedToSameAsset`	boolean	Indicates whether entitlements-related lookup filters on cases return only the entitlements related to the case's asset (`true`) or not (`false`).
`entitlementLookupLimitedToSameContact`	boolean	Indicates whether entitlements-related lookup filters on cases return only the entitlements related to the case's contact (`true`) or not (`false`).
`ignoreMilestoneBusinessHours`	boolean	Indicates whether to show the time remaining on an event milestone in actual hours (`true`) or business hours (`false`). This field is available in API version 47.0 and later.

Declarative Metadata Sample Definition

This is a sample entitlements settings file.

```
<?xml version="1.0" encoding="UTF-8"?>
<EntitlementSettings xmlns="http://soap.sforce.com/2006/04/metadata">
    <assetLookupLimitedToActiveEntitlementsOnAccount>
        false
    </assetLookupLimitedToActiveEntitlementsOnAccount>
    <assetLookupLimitedToActiveEntitlementsOnContact>
        false
    </assetLookupLimitedToActiveEntitlementsOnContact>
    <assetLookupLimitedToSameAccount>
        false
    </assetLookupLimitedToSameAccount>
    <assetLookupLimitedToSameContact>
        false
    </assetLookupLimitedToSameContact>
    <enableEntitlements>
        true
    </enableEntitlements>
    <entitlementLookupLimitedToActiveStatus>
        false
    </entitlementLookupLimitedToActiveStatus>
    <entitlementLookupLimitedToSameAccount>
        false
    </entitlementLookupLimitedToSameAccount>
    <entitlementLookupLimitedToSameAsset>
        false
    </entitlementLookupLimitedToSameAsset>
    <entitlementLookupLimitedToSameContact>
        false
    </entitlementLookupLimitedToSameContact>
</EntitlementSettings>
```

Wildcard Support in the Manifest File

The wildcard character `*` (asterisk) in the `package.xml` manifest file doesn't apply to metadata types for feature settings. The wildcard applies only when retrieving all settings, not for an individual setting. For details, see Settings. For information about using the manifest file, see Deploying and Retrieving Metadata with the Zip File.

EventSettings

Represents an org's platform event settings.

File Suffix and Directory Location

EventSettings components have the suffix `.settings` and are stored in the `settings` folder.

Version

EventSettings components are available in API version 47.0 and later.

Fields

Field Name	Field Type	Description
`enableDeleteMonitoringData`	boolean	Allows (`true`) or disallows (`false`) users to delete event log files and LoginEvent data. Users require the Delete Event Monitoring Records user permission, which is available when this setting is enabled. Default value is `false`.
`enableDynamicStreamingChannel`	boolean	Enables (`true`) or disables (`false`) the dynamic creation of a streaming channel when you subscribe to generic streaming. Default value is `false`.
`enableEventLogWaveIntegration`	boolean	Enables (`true`) or disables (`false`) the integration of event monitoring log files and Analytics apps. Analytics apps help you visualize your user's activity. Default value is `false`.
`enableLoginForensics`	boolean	Enables (`true`) or disables (`false`) the Login Forensics feature. Login Forensics helps you track and audit your org's user login activity. Default value is `false`.
`enableStreamingApi`	boolean	Enables (`true`) or disables (`false`) Streaming API in the org. Default value is `true`.
`enableTerminateOldestSession`	boolean	Determines the behavior of transaction security policies that trigger an end-session action during an API-based login (a login that doesn't come through the UI.) An end-session action occurs when a user exceeds the maximum number of allowed Salesforce sessions. When `true`, and a user triggers an end-session action, Salesforce terminates the user's oldest session until the user is in compliance. When set to `false`, Salesforce blocks the most recent user's attempt to log in and doesn't allow a new user session. Default value is `false`.
`enableTransactionSecurityPolicies`	boolean	Enables (`true`) or disables (`false`) the ability to create and use transaction security policies in the Salesforce UI. Default value is `false`.
`enableApexLimitEvents`	boolean	Enables (`true`) or disables (`false`) the Apex Limit Events (Pilot) feature. Default value is `false`. Note: The Apex Limit Events (Pilot) feature has been discontinued so don't use this field.

Declarative Metadata Sample Definition

The following is an example of an `EventSettings.settings` file.

```
<?xml version="1.0" encoding="UTF-8"?>
<EventSettings xmlns="http://soap.sforce.com/2006/04/metadata">
```

```
    <enableDeleteMonitoringData>true</enableDeleteMonitoringData>
    <enableDynamicStreamingChannel>false</enableDynamicStreamingChannel>
    <enableEventLogWaveIntegration>true</enableEventLogWaveIntegration>
    <enableLoginForensics>true</enableLoginForensics>
    <enableStreamingApi>true</enableStreamingApi>
    <enableTerminateOldestSession>true</enableTerminateOldestSession>
    <enableTransactionSecurityPolicies>true</enableTransactionSecurityPolicies>
</EventSettings>
```

Example Package Manifest

The following is an example package manifest used to deploy or retrieve the Event settings metadata:

```
<?xml version="1.0" encoding="UTF-8"?>
<Package xmlns="http://soap.sforce.com/2006/04/metadata">
   <types>
       <members>Event</members>
       <name>Settings</name>
   </types>
   <version>47.0</version>
</Package>
```

Wildcard Support in the Manifest File

The wildcard character `*` (asterisk) in the `package.xml` manifest file doesn't apply to metadata types for feature settings. The wildcard applies only when retrieving all settings, not for an individual setting. For details, see Settings. For information about using the manifest file, see Deploying and Retrieving Metadata with the Zip File.

ExperienceBundleSettings

Represents the settings of ExperienceBundleMetadata of an organization. This type extends the Metadata metadata type and inherits its `fullName` field.

Note: ExperienceBundleMetadata is a text-based code structure of Community settings and site components, such as pages, branding sets, and themes. Developers can quickly update and deploy one or more Lightning communities programmatically using their preferred development tools. ExperienceBundleSettings represents the org preferences associated with ExperienceBundleMetadata.

File Suffix and Directory Location

ExperienceBundleSettings values are stored in a single file named `ExperienceBundle.settings` in the `settings` directory. The `.settings` files are different from other named components because there is only one settings file for each settings component.

Version

ExperienceBundleSettings is available in API version 46.0 and later.

Fields

Field Name	Field Type	Description
enableExperienceBundleMetadata	boolean	Indicates whether ExperienceBundleMetadata is enabled. Default is false.

Declarative Metadata Sample Definition

Here's an example of ExperienceBundle.settings that references the previous definition.

```
<ExperienceBundleSettings xmlns="http://soap.sforce.com/2006/04/metadata">
        <enableExperienceBundleMetadata>true</enableExperienceBundleMetadata>
</ExperienceBundleSettings>
```

Wildcard Support in the Manifest File

The wildcard character * (asterisk) in the package.xml manifest file doesn't apply to metadata types for feature settings. The wildcard applies only when retrieving all settings, not for an individual setting. For details, see Settings. For information about using the manifest file, see Deploying and Retrieving Metadata with the Zip File.

SEE ALSO:

ExperienceBundle

FieldServiceSettings

Represents an organization's Field Service Lightning settings.

To learn more about Field Service Lightning settings, see Enable Field Service Lightning in the Salesforce Help.

In the package manifest, all organization settings metadata types are accessed using the "Settings" name. See Settings for more details.

Version

FieldServiceSettings is available in API version 40.0 and later.

Fields

Field	Field Type	Description
doesShareSaParentWoWithAr	boolean	Shares service appointments' parent work orders with their assigned resources. This setting applies only if doesShareSaWithAr is selected and sharing access for work orders is set to Private or Public Read Only. Technician assigned resources get Read-Write access to their work orders. For assigned resources of type Crew,

Field	Field Type	Description
		the crew leader gets Read-Write access and crew members get Read access. If the service appointment's parent is a work order line item, assigned resources get access to the associated work order.
`doesShareSaWithAr`	boolean	Shares dispatched service appointments with their assigned resources. This setting applies only if sharing access for service appointments is set to Private or Public Read Only. Technician assigned resources get Read-Write access to their service appointments. For assigned resources of type Crew, the crew leader gets Read-Write access and crew members get Read access.
`enableWorkOrders`	boolean	Enables Work Orders for the org. This setting allows users to use the Work Order object, whether or not Field Service is enabled. When Field Service is enabled, you can't turn off Work Orders.
`isGeoCodeSyncEnabled`	boolean	Syncs the location of a Service Resource to an Inventory object.
`isLocationHistoryEnabled`	boolean	Tracks the location history of a Service Resource.
`fieldServiceNotificationsOrgPref`	boolean	Turns on in-app notifications for the Salesforce app and Lightning Experience users when any of the following actions occurs on a work order or work order line item that they own or follow: • A text or file post is added • A tracked field is updated • The record owner changes • The resource assignments change on a related service appointment If the option to track all related objects is selected in the feed tracking settings for work orders, users are also notified when child records of work orders—such as service appointments—are created or deleted.
`fieldServiceOrgPref`	boolean	Indicates whether Field Service Lightning is enabled.

Field	Field Type	Description
`serviceAppointmentsDueDateOffsetOrgValue`	int	Indicates the number of days past the Created Date that the Due Date on auto-created service appointments should fall. Work types include an option to automatically add a service appointment to new work orders or work order line items using the work type.
`workOrderLineItemSearchFields`	string	The work order line item fields that the search engine should scan to suggest knowledge articles on work order line items.
`workOrderSearchFields`	string	The work order fields that the search engine should scan to suggest knowledge articles on work orders.

Declarative Metadata Sample Definition

This sample file shows a subset of all of the possible field service settings that you can customize.

```
<?xml version="1.0" encoding="UTF-8"?>
<FieldServiceSettings xmlns="http://soap.sforce.com/2006/04/metadata">
    <fieldServiceNotificationsOrgPref>false</fieldServiceNotificationsOrgPref>
    <fieldServiceOrgPref>true</fieldServiceOrgPref>
   <serviceAppointmentsDueDateOffsetOrgValue>6</serviceAppointmentsDueDateOffsetOrgValue>

    <enableWorkOrders>true</enableWorkOrders>
    <workOrderSearchFields>WorkOrderNumber</workOrderSearchFields>
    <workOrderSearchFields>Status</workOrderSearchFields>
    <workOrderSearchFields>Address</workOrderSearchFields>
    <workOrderSearchFields>Subject</workOrderSearchFields>
    <workOrderLineItemSearchFields>Subject</workOrderLineItemSearchFields>
</FieldServiceSettings>
```

Wildcard Support in the Manifest File

The wildcard character `*` (asterisk) in the `package.xml` manifest file doesn’t apply to metadata types for feature settings. The wildcard applies only when retrieving all settings, not for an individual setting. For details, see Settings. For information about using the manifest file, see Deploying and Retrieving Metadata with the Zip File.

FileUploadAndDownloadSecuritySettings

Represents the security settings for uploading and downloading files. This type extends the Metadata metadata type and inherits its `fullName` field.

File Suffix and Directory Location

FileUploadAndDownloadSecuritySettings components have the suffix `.settings` and are stored in the `settings` folder.

Version

FileUploadAndDownloadSecuritySettings components are available in API version 39.0 and later.

Fields

Field Name	Field Type	Description
`dispositions`	FileTypeDispositionAssignmentBean[]	Represents the metadata used to manage filetype behavior. This field is available in API version 39.0 and later.
`noHtmlUploadAsAttachment`	boolean	Indicates whether to allow HTML uploads as attachments or document records. This field is available in API version 39.0 and later.

FileTypeDispositionAssignmentBean

Represents the metadata used to manage filetype behavior.

Field Name	Field Type	Description
`behavior`	FileDownloadBehavior (enumeration of type string)	One of the following values: • DOWNLOAD • EXECUTE • HYBRID The following filetypes are a security risk and can not have EXECUTE behavior: • EXE • FLASH • HTML • RFC822 • SVG • TXML • UNKNOWN • WEBVIEW • XHTML • XML
`filetype`	FileType (enumeration of type string)	Although more filetypes exist, these are the only ones supported by FileTypeDispositionAssignmentBean: • AVI • EXCEL • EXCEL_X • EXE

Field Name	Field Type	Description
		• FLASH • HTML • MOV • MP3 • MP4 • MPEG • PDF • POWER_POINT • POWER_POINT_X • RFC822 • SVG • TXML • UNKNOWN • WAV • WEBVIEW • WMA • WMV • WORD • WORD_X • XHTML • XML
securityRiskFileType	boolean	Indicates filetypes that cannot have behavior set to EXECUTE, due to security risks. This field is read-only.

Declarative Metadata Sample Definition

The following is an example of a FileUploadAndDownloadSecuritySettings component.

```
<FileUploadAndDownloadSecuritySettings xmlns="http://soap.sforce.com/2006/04/metadata">
    <dispositions>
        <behavior>HYBRID</behavior>
        <fileType>AVI</fileType>
        <securityRiskFileType>false</securityRiskFileType>
    </dispositions>
    <dispositions>
        <behavior>HYBRID</behavior>
        <fileType>WORD</fileType>
        <securityRiskFileType>false</securityRiskFileType>
    </dispositions>
    <dispositions>
        <behavior>HYBRID</behavior>
        <fileType>WORD_X</fileType>
        <securityRiskFileType>false</securityRiskFileType>
    </dispositions>
    <dispositions>
        <behavior>DOWNLOAD</behavior>
        <fileType>EXE</fileType>
        <securityRiskFileType>true</securityRiskFileType>
    </dispositions>
    <dispositions>
        <behavior>DOWNLOAD</behavior>
        <fileType>HTML</fileType>
        <securityRiskFileType>true</securityRiskFileType>
    </dispositions>
    <dispositions>
        <behavior>DOWNLOAD</behavior>
        <fileType>WEBVIEW</fileType>
        <securityRiskFileType>true</securityRiskFileType>
    </dispositions>
    <dispositions>
        <behavior>DOWNLOAD</behavior>
        <fileType>RFC822</fileType>
        <securityRiskFileType>true</securityRiskFileType>
    </dispositions>
    <dispositions>
        <behavior>HYBRID</behavior>
        <fileType>MOV</fileType>
        <securityRiskFileType>false</securityRiskFileType>
    </dispositions>
    <dispositions>
        <behavior>HYBRID</behavior>
        <fileType>MP3</fileType>
        <securityRiskFileType>false</securityRiskFileType>
    </dispositions>
    <dispositions>
        <behavior>HYBRID</behavior>
        <fileType>MP4</fileType>
        <securityRiskFileType>false</securityRiskFileType>
    </dispositions>
    <dispositions>
        <behavior>HYBRID</behavior>
        <fileType>MPEG</fileType>
        <securityRiskFileType>false</securityRiskFileType>
    </dispositions>
    <dispositions>
        <behavior>HYBRID</behavior>
        <fileType>PDF</fileType>
        <securityRiskFileType>false</securityRiskFileType>
    </dispositions>
    <dispositions>
        <behavior>HYBRID</behavior>
        <fileType>POWER_POINT</fileType>
        <securityRiskFileType>false</securityRiskFileType>
    </dispositions>
    <dispositions>
        <behavior>HYBRID</behavior>
        <fileType>POWER_POINT_X</fileType>
        <securityRiskFileType>false</securityRiskFileType>
    </dispositions>
    <dispositions>
        <behavior>DOWNLOAD</behavior>
        <fileType>SVG</fileType>
        <securityRiskFileType>true</securityRiskFileType>
    </dispositions>
    <dispositions>
        <behavior>DOWNLOAD</behavior>
        <fileType>FLASH</fileType>
        <securityRiskFileType>true</securityRiskFileType>
    </dispositions>
    <dispositions>
        <behavior>DOWNLOAD</behavior>
        <fileType>TXML</fileType>
        <securityRiskFileType>true</securityRiskFileType>
    </dispositions>
    <dispositions>
        <behavior>DOWNLOAD</behavior>
        <fileType>UNKNOWN</fileType>
        <securityRiskFileType>true</securityRiskFileType>
    </dispositions>
    <dispositions>
        <behavior>HYBRID</behavior>
        <fileType>WAV</fileType>
        <securityRiskFileType>false</securityRiskFileType>
    </dispositions>
    <dispositions>
        <behavior>HYBRID</behavior>
        <fileType>WMA</fileType>
        <securityRiskFileType>false</securityRiskFileType>
    </dispositions>
    <dispositions>
        <behavior>HYBRID</behavior>
        <fileType>WMV</fileType>
        <securityRiskFileType>false</securityRiskFileType>
    </dispositions>
    <dispositions>
        <behavior>DOWNLOAD</behavior>
        <fileType>XHTML</fileType>
        <securityRiskFileType>true</securityRiskFileType>
    </dispositions>
    <dispositions>
        <behavior>HYBRID</behavior>
        <fileType>EXCEL</fileType>
        <securityRiskFileType>false</securityRiskFileType>
    </dispositions>
    <dispositions>
        <behavior>HYBRID</behavior>
        <fileType>EXCEL_X</fileType>
        <securityRiskFileType>false</securityRiskFileType>
    </dispositions>
    <dispositions>
        <behavior>DOWNLOAD</behavior>
        <fileType>XML</fileType>
        <securityRiskFileType>true</securityRiskFileType>
    </dispositions>
    <noHtmlUploadAsAttachment>false</noHtmlUploadAsAttachment>
</FileUploadAndDownloadSecuritySettings>
```

The following is an example `package.xml` that references the previous definition.

```
<?xml version="1.0" encoding="UTF-8"?>
<Package xmlns="http://soap.sforce.com/2006/04/metadata">
   <types>
       <members>FileUploadAndDownloadSecurity</members>
       <name>Settings</name>
   </types>
   <version>39.0</version>
</Package>
```

Wildcard Support in the Manifest File

The wildcard character `*` (asterisk) in the `package.xml` manifest file doesn't apply to metadata types for feature settings. The wildcard applies only when retrieving all settings, not for an individual setting. For details, see Settings. For information about using the manifest file, see Deploying and Retrieving Metadata with the Zip File.

FlowSettings

Represents the org's settings for processes and flows, such as whether Lightning runtime for flows is enabled. This type extends the Metadata metadata type and inherits its `fullName` field.

In the package manifest, all organization settings metadata types are accessed using the "Settings" name. See Settings for more details.

File Suffix and Directory Location

FlowSettings values are stored in the `Flow.settings` file in the settings directory. The `.settings` files are different from other named components because there is only one settings file for each settings component.

Version

FlowSettings components are available in API version 47.0 and later.

Fields

Field Name	Field Type	Description
`doesFormulaEnforceDataAccess`	boolean	Indicates whether formula resources and formula fields in a flow enforce record-level security (`true`) or not (`false`). Corresponds to the Enforce Data Access in Flow Formulas critical update. Available in API version 48.0 and later.
`doesFormulaGenerateHtmlOutput`	boolean	Reserved for future use.

Field Name	Field Type	Description
`enableFlowBREncodedFixEnabled`	boolean	Indicates whether `BR()` functions in flow and process formulas result in a line break (`true`) or resolve to `_BR_ENCODED_` as a literal value (`false`). Corresponds to the Use the BR() Function in Flows and Processes critical update.
`enableFlowDeployAsActiveEnabled`	boolean	Indicates whether processes and flows can be deployed as active via change sets or Metadata API. When the value is `false`, all processes and flows are deployed as inactive. When the value is `true`, deploying an active process or flow in a production org causes your Apex tests to run. If Apex tests don't launch your org's required percentage of active processes and autolaunched flows, the deployment is rolled back. The default value is `false` for production orgs and is `true` for non-production orgs such as scratch, sandbox, and developer orgs. Corresponds to the `Deploy processes and flows as active` field on the Process Automation Settings page in Setup. The field appears in the user interface on production orgs only.
`enableFlowFieldFilterEnabled`	boolean	Indicates whether flows can successfully execute Create Records and Update Records elements that update fields to which the running doesn't have edit access. By default (`false`), the Create Records or Update Records element fails and executes the fault path if it has one. When the value is `true`, the element sets only the fields that the running user can edit. No notification is sent when some fields aren't updated. Corresponds to the `Filter inaccessible fields from flow requests` field on the Process Automation Settings page in Setup.
`enableFlowFormulasFixEnabled`	boolean	Indicates whether process and flow formulas return null values when the calculations involve a null record variable or null lookup relationship field. When the value is `true`, those formulas return null values at run time. When the value is `false`, those formulas return unhandled exceptions at run time. Corresponds to the Check for Null Record Variables or Null Values of Lookup Relationship Fields in Process and Flow Formulas critical update.
`enableFlowInterviewSharingEnabled`	boolean	Indicates whether users can resume the paused flow interviews that they have edit access to. By default (`true`), users can resume interviews that are shared with them, either directly or via the role hierarchy. When the value is `false`, each paused interview can be resumed only by the interview owner or a flow admin who has view access to the interview. Corresponds to the `Let users resume shared flow interviews` field on the Process Automation Settings page in Setup.

Field Name	Field Type	Description
`enableFlowNullPreviousValueFix`	boolean	Indicates whether each process evaluates criteria by always using the original record field values from when the process begins. When the value is `true`, each process with an Update Records action and multiple criteria nodes always evaluates criteria using the original field values of the record. When the value is `false`, processes evaluate the updated values of record fields that were null when the process began. Corresponds to the Evaluate Criteria Based on Original Record Values in Process Builder critical update.
`enableFlowPauseEnabled`	boolean	Indicates whether screens can display the Pause button so that users can pause flow interviews. By default, the value is `false`. Corresponds to the `Let users pause flows` field on the Process Automation Settings page in Setup.
`enableFlowUseApexExceptionEmail`	boolean	Indicates whether process and flow error emails are sent to: • The user who last modified the process or flow (`false`) • The addresses set on the Apex Exception Email page in Setup (`true`) By default, the value is `false`. Corresponds to the `Send Process or Flow Error Email to` field on the Process Automation Settings page in Setup.
`enableInvocableFlowFixEnabled`	boolean	Indicates whether all autolaunched flow interviews are executed when they're invoked in bulk from a process or the Invocable Actions resource in REST API (`true`) or not (`false`). When the value is `false`, flow interviews that share identical input parameters are not executed. Corresponds to the Execute All Flow Interviews When Invoked in Bulk critical update.
`enableLightningRuntimeEnabled`	boolean	Indicates whether flows that are launched from a URL or from Setup use the Lightning runtime experience (`true`) or the Classic runtime experience (`false`). By default, the value is `true`. Corresponds to the `Enable Lightning runtime for flows` field on the Process Automation Settings page in Setup.
`isAccessToInvokedApexRequired`	boolean	Indicates whether flows can invoke Apex classes only when the running users' profiles or permission sets include access to those Apex classes. When the value is `false`, Apex class security doesn't apply to flows. Corresponds to the Require User Access to Apex Classes Invoked by Flow critical update.
`isApexPluginAccessModifierRespected`	boolean	Indicates whether flows respect the public access modifiers for legacy Apex actions. When the value is `true`: • Flows fail when they execute public legacy Apex actions from a different namespace.

Field Name	Field Type	Description
		- Public legacy Apex actions from a different namespace aren't available in Flow Builder. - Global legacy Apex actions with public `describe` or `invoke` methods are unavailable to flows in a different namespace. When the value is `false`, you can add public legacy Apex actions to flows even though they're not supported. Also, global legacy Apex actions with public `describe` or `invoke` methods are available to flows in a different namespace. Corresponds to the Make Flows Respect Access Modifiers for Legacy Apex Actions critical update. Available in API version 48.0 and later.
`isEnhancedFlowListViewVisible`	boolean	Indicates whether the enhanced Flows list view in Lightning Experience replaces the Classic Flows list view (`true`) or not (`false`). The default value is `true`. If the field is set to `false`, the Classic Flows list view replaces the enhanced list view.
`isManageFlowRequiredForAutomationCharts`	boolean	Indicates whether the Manage Flow permission is required to view all charts in Automation Home (Beta) (`true`) or not (`false`). The default value is `false`. All users with the View Setup and Configuration permission can view all charts in Automation Home. If the field is set to `true`, then users with the View Setup and Configuration permission can view only the Total Started Automations by Process Type chart. The Manage Flow permission is required to view all charts.

Declarative Metadata Sample Definition

The following is an example of the Flow.settings file.

```
<?xml version="1.0" encoding="UTF-8"?>
<FlowSettings xmlns="http://soap.sforce.com/2006/04/metadata">
    <doesFormulaEnforceDataAccess>true</doesFormulaEnforceDataAccess>
    <enableFlowBREncodedFixEnabled>true</enableFlowBREncodedFixEnabled>
    <enableFlowDeployAsActiveEnabled>false</enableFlowDeployAsActiveEnabled>
    <enableFlowFieldFilterEnabled>false</enableFlowFieldFilterEnabled>
    <enableFlowFormulasFixEnabled>true</enableFlowFormulasFixEnabled>
    <enableFlowInterviewSharingEnabled>true</enableFlowInterviewSharingEnabled>
    <enableFlowNullPreviousValueFix>true</enableFlowNullPreviousValueFix>
    <enableFlowPauseEnabled>true</enableFlowPauseEnabled>
    <enableFlowUseApexExceptionEmail>false</enableFlowUseApexExceptionEmail>
    <enableInvocableFlowFixEnabled>true</enableInvocableFlowFixEnabled>
    <enableLightningRuntimeEnabled>true</enableLightningRuntimeEnabled>
    <isAccessToInvokedApexRequired>true</isAccessToInvokedApexRequired>
    <isApexPluginAccessModifierRespected>true</isApexPluginAccessModifierRespected>
    <isEnhancedFlowListViewVisible>true</isEnhancedFlowListViewVisible>
   <isManageFlowRequiredForAutomationCharts>false</isManageFlowRequiredForAutomationCharts>
</FlowSettings>
```

Example Package Manifest

The following is an example package.xml that references the previous definition.

```
<?xml version="1.0" encoding="UTF-8"?>
<Package xmlns="http://soap.sforce.com/2006/04/metadata">
 <types>
  <members>Flow</members>
  <name>Settings</name>
 </types>
 <version>47.0</version>
</Package>
```

Wildcard Support in the Manifest File

The wildcard character * (asterisk) in the package.xml manifest file doesn't apply to metadata types for feature settings. The wildcard applies only when retrieving all settings, not for an individual setting. For details, see Settings. For information about using the manifest file, see Deploying and Retrieving Metadata with the Zip File.

ForecastingSettings

Represents the Collaborative Forecasts settings options. This type extends the Metadata metadata type and inherits its fullName field.

Note: This information only applies to Collaborative Forecasts.

In the package manifest, all organization settings metadata types are accessed using the "Settings" name. See Settings for more details.

File Suffix and Directory Location

ForecastingSettings values are stored in a single file named Forecasting.settings in the settings directory of the corresponding package directory. The .settings files are different from other named components because there is only one settings file for each settings component.

Version

ForecastingSettings components are available in API version 28 and later. The structure of the ForecastingSettings type changed significantly in API version 30.0.

Fields

Field Name	Field Type	Description
displayCurrency	DisplayCurrency (enumeration of type string)	Removed. The currency for displaying forecasts; either the organization's corporate currency or each forecast owner's personal currency setting. The selected currency is the default used in Collaborative Forecasts and selected in setup. The selection must be one of the currencies enabled for use in the organization, and only one selection is allowed. The default is CORPORATE. The valid values are: • CORPORATE

Field Name	Field Type	Description
		- `PERSONAL` Available in API version 28.0 to 46.0. In API version 47.0 and later, use `defaultToPersonalCurrency`.
`defaultToPersonalCurrency`	boolean	If multicurrency is enabled, this field indicates whether the user's personal currency is used in forecasts. If `true` (default), the user's personal currency is used. If `false`, the corporate currency is used. Available in API version 47.0 and later.
`enableForecasts`	boolean	Indicates if Collaborative Forecasts is enabled or not. Set to `true` to enable Collaborative Forecasts and `false` to disable the functionality. Warning: Disabling Forecasts can result in data loss. Refer to Salesforce Help before disabling any functionality.
`forecastingCategoryMappings`	ForecastingCategoryMapping[]	A list of mappings associating forecast types with forecast rollups. As of Spring '20 and later, only standard users with the View All Forecasts or Allow Forecasting permission or delegated forecast manager status can access this subtype.
`forecastingDisplayedFamilySettings`	ForecastingDisplayedFamilySettings[]	The product families chosen to allow forecasting on in Lightning Experience. This field is available in API version 40.0 and later.
`forecastingTypeSettings`	ForecastingTypeSettings[]	A list of forecast types. For field values, see ForecastingTypeSettings. The maximum number of forecast types is four.

ForecastingCategoryMapping

The forecasting category mappings for Collaborative Forecasts. This subtype appears eight times within the `ForecastingSettings` type. Each occurrence includes fields that specify a type of forecast category rollup, which forecast categories each rollup includes, and the weight of each forecast category in the rollup. Organizations using either cumulative forecast rollups or individual forecast category columns must include all eight occurrences of this subtype. As of Spring '20 and later, only standard users with the View All Forecasts or Allow Forecasting permission or delegated forecast manager status can access this subtype.

Field	Field Type	Description
`forecastingItemCategoryApiName`	string	Required. This field specifies the API name of the rollup type. The valid values are: - `openpipeline` - `bestcaseforecast` - `commitforecast` - `pipelineonly` - `bestcaseonly` - `commitonly` - `closedonly` - `omittedonly`

Field	Field Type	Description
`weightedSourceCategories`	WeightedSourceCategory[]	This field can occur more than once when specifying more than one forecast category to include in the rollup type. Each occurrence contains two subfields that specify a forecast category to include in the forecast rollup type and its weight. Some rollup types include more than one forecast category. This list shows the forecast categories that are included in each rollup type. • Rollup: `openpipeline`, Forecast categories: pipeline, best case, commit • Rollup: `bestcaseforecast`, Forecast categories: best case, commit, closed • Rollup: `commitforecast`, Forecast categories: commit, closed • Rollup: `pipelineonly`, Forecast categories: `pipeline` • Rollup: `bestcaseonly`, Forecast categories: `best case` • Rollup: `commitonly`, Forecast categories: `commit` • Rollup: `closedonly`, Forecast categories: `closed` • Rollup: `omittedonly`, Forecast categories: `omitted`

ForecastingDisplayedFamilySettings

The product families that an admin chooses to allow forecasting on in Lightning Experience. This field is available in API version 40.0 and later.

Field	Field Type	Description
`productFamily`	string	The product family available to forecast on. Each product family is unique.

ForecastingTypeSettings

The settings for each forecast type. An organization can have up to four forecast types active. If you omit a previously enabled forecast type that has a minimum API version less than or equal to the metadata package version, its quota and adjustment data is deleted from the org.

Warning: Omitting a forecast type field from the XML can deactivate that forecast type: if the forecast type was available in the release specified by the XML package version, that forecast type is deactivated and its quota and adjustment data are deleted.

Field Name	Field Type	Description
active	boolean	Required. Indicates whether the forecast type specified in the name field is active. Note: Setting the active field to false purges all forecasting data, adjustments, and quotas for the forecast type. When active is set to true, some values on the Forecasts tab might not appear immediately. An in-process icon appears to indicate that the values are being calculated.
adjustmentsSettings	AdjustmentsSettings	Required. This enables or disables the Forecasts adjustments option in Forecasts.
displayedCategoryApiNames	string	This read-only field appears four times to specify the four forecast rollup categories displayed in the Forecasts tab, for either cumulative forecast rollups, or individual forecast category rollups. Always use the same 4 values for both displayedCategoryApiNames and forecastedCategoryApiNames. Valid values for organizations using cumulative forecast rollups: • openpipeline • bestcaseforecast • commitforecast • closedonly Valid values for organizations using individual forecast category rollups: • pipelineonly • bestcaseonly • commitonly • closedonly
forecastedCategoryApiNames	string	This field appears four times to specify the four forecast rollup categories used in the organization, for either cumulative forecast rollups, or individual forecast category rollups. Valid values for organizations using cumulative forecast rollups: • openpipeline • bestcaseforecast • commitforecast • closedonly Valid values for organizations using individual forecast category rollups: • pipelineonly • bestcaseonly • commitonly • closedonly

Field Name	Field Type	Description
		Changing from one set of four values to the other changes the organization setting for Enable Cumulative Forecast Rollups in Setup. If this field is omitted, the setting is not changed.
`ForecastingDateType`	ForecastingDateType (enumeration of type string)	Required. The date type that forecast amounts are based on in Collaborative Forecasts. Valid values are: • `OpportunityCloseDate` (default) • `ProductDate` • `ScheduleDate` Available in API version 42.0 and later. In API version 42.0 only, date types are read only and available only via API.
`forecastRangeSettings`	ForecastRangeSettings	Required. The default periods and range selections in Collaborative Forecasts.
`hasProductFamily`	boolean	Required. Indicates whether the forecasting type has product family forecasts enabled. Available in API version 41.0 and later.
`isAmount`	boolean	Required. This read-only field indicates whether the forecast type is based on revenue amounts. The value of `isAmount` is always the opposite of the value of `isQuantity`.
`isAvailable`	boolean	Required. This read-only field indicates whether the forecast type can currently be used in the organization. For example, the revenue splits forecast type can't be used in an organization that doesn't have Opportunity Splits enabled.
`isQuantity`	boolean	Required. This read-only field indicates whether the forecast type is based on product quantities. The value of `isQuantity` is always the opposite of the value of `isAmount`.
`managerAdjustableCategoryApiNames`	string	This read-only field appears twice to specify the two forecast rollup categories that forecast managers can adjust in the organization for either cumulative forecast rollups or individual forecast category rollups. This field can only be used when the `enableAdjustments` field contains a value of true. If both the `managerAdjustableCategoryApiNames` and `ownerAdjustableCategoryApiNames` fields are being used, they must contain the same two values. Their values must also be consistent with the values of the `enableAdjustments` and `enableOwnerAdjustments` fields. Valid values for organizations using cumulative forecast rollups: • `bestcaseforecast` • `commitforecast` Valid values for organizations using individual forecast category rollups:

Field Name	Field Type	Description
		• `bestcaseonly` • `commitonly`
`masterLabel`	string	Required. This read-only field indicates the UI label for the forecast type.
`name`	string	Required. The name of the forecast type. Each forecast type requires a specific string. Valid values include: • `LineItemQuantityProductDate` : Product Families - Quantity by product date. Available in API version 47.0 and later. • `LineItemQuantityScheduleDate` : Product Families - Quantity by schedule date. Available in API version 47.0 and later. • `LineItemRevenueProductDate` : Product Families - Revenue by product date. Available in API version 47.0 and later. • `LineItemRevenueScheduleDate` : Product Families - Revenue by schedule date. Available in API version 47.0 and later. • `OpportunityLineItemQuantity` : Product Families - Quantity • `OpportunityLineItemRevenue` : Product Families - Revenue • `OpportunityOverlayRevenue` : Opportunity Overlay Splits - Revenue • `OpportunityQuantity` : Opportunities - Quantity • `OpportunityQuantityProductDate` : Opportunities - Quantity by product date. Available in API version 43.0 and later. • `OpportunityQuantityScheduleDate` : Opportunities - Quantity by schedule date. Available in API version 43.0 and later. • `OpportunityRevenue` : Opportunities - Revenue • `OpportunityRevenueProductDate` : Opportunities - Revenue by product date. Available in API version 43.0 and later. • `OpportunityRevenueScheduleDate` : Opportunities - Revenue by schedule date. Available in API version 43.0 and later. • `OpportunitySplitRevenue` : Opportunity Revenue Splits - Revenue • ***`Territory_Model_NameN`***: Territories, where *`Territory_Model_Name`* is the name of your active territory model in the API. *`Territory_Model_Name`* can be followed by *`N`*, an auto-generated number that distinguishes between territory forecast types. Available in API version 44.0 and later. • ***`Territory_Model_NameN`***`_ProductFamily`: Deprecated. Territories - Product Families, where *`Territory_Model_Name`* is the name of your active territory model in the API and can be followed by *`N`*, an auto-generated number that distinguishes

Field Name	Field Type	Description
		between territory forecast types. Available in API version 45.0 and 46.0. For territory models created in API version 47.0 and later, ***Territory_Model_NameN*** is used. - The name of a custom opportunity split type that has been enabled as a forecast type. Custom split types are based on currency fields, which can contain revenue amounts only.
opportunityListFieldsLabelMappings	OpportunityListFieldsLabelMapping	A read-only list of the API names and UI labels for all fields on the Opportunity object.
opportunityListFieldsSelectedSettings	OpportunityListFieldsSelectedSettings	Required. The fields selected to appear in the opportunity pane of the forecast page for the forecast type. One of the selected fields must be **Opportunity Name**. You can select up to 15 fields.
opportunityListFieldsUnselectedSettings	OpportunityListFieldsUnselectedSettings	Required. The fields *not* selected to appear in the opportunity pane of the forecast page for the forecast type.
opportunitySplitName	string	Indicates whether the forecasting type has a split type, and if so, the name of the split type. Available in API version 41.0 and later.
ownerAdjustableCategoryApiNames	string	This read-only field appears twice to specify the two forecast rollup categories that forecast owners can adjust in the organization, for either cumulative forecast rollups, or individual forecast category rollups. This field can only be used when the enableOwnerAdjustments field contains a value of true. If both the managerAdjustableCategoryApiNames and ownerAdjustableCategoryApiNames fields are being used, they must contain the same two values. Their values must also be consistent with the values of the enableAdjustments and enableOwnerAdjustments fields. Valid values for organizations using cumulative forecast rollups: - bestcaseforecast - commitforecast Valid values for organizations using individual forecast category rollups: - bestcaseonly - commitonly
quotasSettings	QuotasSettings	Required. This enables or disables the quota option in Forecasts.
territory2ModelName	string	Indicates whether the forecasting type has a Territory2 model, and if so, the name of the Territory2 model. Available in API version 41.0 and later.

AdjustmentsSettings

The adjustment options for Collaborative Forecasts.

Field	Field Type	Description
`enableAdjustments`	boolean	Required. Set to `true` to enable Collaborative Forecasts manager adjustments and `false` to disable them. All forecast types must contain the same `enableAdjustments` value. Warning: Disabling adjustments results in Collaborative Forecasts adjustment data being purged.
`enableOwnerAdjustments`	boolean	Required. Set to `true` to enable Collaborative Forecasts owner adjustments and `false` to disable them. All forecast types must contain the same `enableAdjustments` value. Warning: Disabling adjustments results in Collaborative Forecasts adjustment data being purged.

ForecastRangeSettings

The default periods and range selections in Collaborative Forecasts. Users can forecast up to 15 months, 15 fiscal periods, or 8 quarters in the future or past. If your forecast range includes the current month, period, or quarter, the forecasts page displays the current month, period, or quarter by default. If not, the first month, period, or quarter of the range is selected. All forecast types must contain the same `forecastRangeSettings` field values.

Warning: If you change the time period from monthly to quarterly or quarterly to monthly, or you change the standard fiscal year, all adjustments and quotas are purged. If you enable custom fiscal years, creating the first custom fiscal year deletes any quotas and adjustments in the corresponding and subsequent standard fiscal years. These changes trigger a forecast recalculation that can take significant time, depending on the quantity of your data.

Field	Field Type	Description
`beginning`	int	Required. Indicates the beginning month or quarter to display by default.
`displaying`	int	Required. Indicates the number of months or quarters to display by default. The maximum number of months is 12 and quarters is 8.
`periodType`	PeriodTypes (enumeration of type string)	Required. Indicates what type of period to use. Valid values are: - `Month` - `Quarter` - `Week` - `Year`

OpportunityListFieldsLabelMapping

A read-only list of the API names and UI labels for all fields on the Opportunity object.

Field	Field Type	Description
`field`	string	Required. The API name of the Opportunity field.
`label`	string	Required. The UI label of the Opportunity field.

OpportunityListFieldsSelectedSettings

The fields selected to appear in the opportunity pane of the forecast page for the forecast type. One of the selected fields must be **Opportunity Name**. You can select up to 15 fields.

Field	Field Type	Description
`field`	string	Specifies names of fields to display in the opportunity pane.

OpportunityListFieldsUnselectedSettings

The fields *not* selected to appear in the opportunity pane of the forecast page for the forecast type.

Field	Field Type	Description
`field`	string	Specifies names of fields not displayed in the opportunity pane.

QuotasSettings

QuotasSettings indicates if quotas are available in Collaborative Forecasts.

Field	Field Type	Description
`showQuotas`	boolean	Required. Set to `true` to enable quotas. All forecast types must contain the same `showQuotas` field value.

WeightedSourceCategory

This field can occur more than once when specifying more than one forecast category to include in the rollup type. Each occurrence contains two subfields that specify a forecast category to include in the forecast rollup type and its weight. Some rollup types include more than one forecast category. This table shows the forecast categories that are included in each rollup type.

Field	Field Type	Description
`sourceCategoryApiName`	string	Required. Specifies the API name of a forecast category to include in the rollup type. The valid values are. • `pipeline` • `best case` • `commit` • `closed` • `omitted`

Field	Field Type	Description
`weight`	double	Required. Specifies the weight given to the forecast category when calculating the forecast for the rollup type. The only supported value is `1.0`.

Declarative Metadata Sample Definition

The following is an example of a ForecastingSettings component that enables the Opportunity-Revenue and Product Family-Quantity forecast types:

```
<?xml version="1.0" encoding="UTF-8"?>
<ForecastingSettings xmlns="http://soap.sforce.com/2006/04/metadata">
    <enableForecasts>true</enableForecasts>
    <forecastingTypeSettings>
        <active>true</active>
        <adjustmentsSettings>
            <enableAdjustments>true</enableAdjustments>
        </adjustmentsSettings>
        <name>OpportunityRevenue</name>
        <forecastRangeSettings>
            <beginning>0</beginning>
            <displaying>6</displaying>
            <periodType>Month</periodType>
        </forecastRangeSettings>
        <opportunityListFieldsSelectedSettings>
            <field>OPPORTUNITY.NAME</field>
        </opportunityListFieldsSelectedSettings>
        <quotasSettings>
            <showQuotas>true</showQuotas>
        </quotasSettings>
    </forecastingTypeSettings>
    <forecastingTypeSettings>
        <active>false</active>
        <adjustmentsSettings>
            <enableAdjustments>true</enableAdjustments>
        </adjustmentsSettings>
        <name>OpportunityLineItemQuantity</name>
        <forecastRangeSettings>
            <beginning>0</beginning>
            <displaying>6</displaying>
            <periodType>Month</periodType>
        </forecastRangeSettings>
        <opportunityListFieldsSelectedSettings>
            <field>OPPORTUNITY.NAME</field>
        </opportunityListFieldsSelectedSettings>
        <quotasSettings>
            <showQuotas>true</showQuotas>
        </quotasSettings>
        <displayedCategoryApiNames>pipelineonly</displayedCategoryApiNames>
        <displayedCategoryApiNames>bestcaseonly</displayedCategoryApiNames>
        <displayedCategoryApiNames>commitonly</displayedCategoryApiNames>
        <displayedCategoryApiNames>closedonly</displayedCategoryApiNames>
        <forecastedCategoryApiNames>commitonly</forecastedCategoryApiNames>
        <forecastedCategoryApiNames>closedonly</forecastedCategoryApiNames>
        <forecastedCategoryApiNames>bestcaseonly</forecastedCategoryApiNames>
        <forecastedCategoryApiNames>pipelineonly</forecastedCategoryApiNames>
        <managerAdjustableCategoryApiNames>commitonly</managerAdjustableCategoryApiNames>

       <managerAdjustableCategoryApiNames>bestcaseonly</managerAdjustableCategoryApiNames>

        <ownerAdjustableCategoryApiNames>commitonly</ownerAdjustableCategoryApiNames>
        <ownerAdjustableCategoryApiNames>bestcaseonly</ownerAdjustableCategoryApiNames>
    </forecastingTypeSettings>
    <forecastingCategoryMappings>
        <forecastingItemCategoryApiName>commitonly</forecastingItemCategoryApiName>
        <weightedSourceCategories>
            <sourceCategoryApiName>commit</sourceCategoryApiName>
            <weight>1.0</weight>
        </weightedSourceCategories>
    </forecastingCategoryMappings>
    <forecastingCategoryMappings>
        <forecastingItemCategoryApiName>closedonly</forecastingItemCategoryApiName>
        <weightedSourceCategories>
            <sourceCategoryApiName>closed</sourceCategoryApiName>
            <weight>1.0</weight>
        </weightedSourceCategories>
    </forecastingCategoryMappings>
    <forecastingCategoryMappings>
        <forecastingItemCategoryApiName>bestcaseforecast</forecastingItemCategoryApiName>

        <weightedSourceCategories>
            <sourceCategoryApiName>commit</sourceCategoryApiName>
            <weight>1.0</weight>
        </weightedSourceCategories>
        <weightedSourceCategories>
            <sourceCategoryApiName>best case</sourceCategoryApiName>
            <weight>1.0</weight>
        </weightedSourceCategories>
        <weightedSourceCategories>
            <sourceCategoryApiName>closed</sourceCategoryApiName>
            <weight>1.0</weight>
        </weightedSourceCategories>
    </forecastingCategoryMappings>
    <forecastingCategoryMappings>
        <forecastingItemCategoryApiName>omittedonly</forecastingItemCategoryApiName>
        <weightedSourceCategories>
            <sourceCategoryApiName>omitted</sourceCategoryApiName>
            <weight>1.0</weight>
        </weightedSourceCategories>
    </forecastingCategoryMappings>
    <forecastingCategoryMappings>
        <forecastingItemCategoryApiName>openpipeline</forecastingItemCategoryApiName>
        <weightedSourceCategories>
            <sourceCategoryApiName>commit</sourceCategoryApiName>
            <weight>1.0</weight>
        </weightedSourceCategories>
        <weightedSourceCategories>
            <sourceCategoryApiName>best case</sourceCategoryApiName>
            <weight>1.0</weight>
        </weightedSourceCategories>
        <weightedSourceCategories>
            <sourceCategoryApiName>pipeline</sourceCategoryApiName>
            <weight>1.0</weight>
        </weightedSourceCategories>
    </forecastingCategoryMappings>
    <forecastingCategoryMappings>
        <forecastingItemCategoryApiName>bestcaseonly</forecastingItemCategoryApiName>
        <weightedSourceCategories>
            <sourceCategoryApiName>best case</sourceCategoryApiName>
            <weight>1.0</weight>
        </weightedSourceCategories>
    </forecastingCategoryMappings>
    <forecastingCategoryMappings>
        <forecastingItemCategoryApiName>commitforecast</forecastingItemCategoryApiName>
        <weightedSourceCategories>
            <sourceCategoryApiName>closed</sourceCategoryApiName>
            <weight>1.0</weight>
        </weightedSourceCategories>
        <weightedSourceCategories>
            <sourceCategoryApiName>commit</sourceCategoryApiName>
            <weight>1.0</weight>
        </weightedSourceCategories>
    </forecastingCategoryMappings>
    <forecastingCategoryMappings>
        <forecastingItemCategoryApiName>pipelineonly</forecastingItemCategoryApiName>
        <weightedSourceCategories>
            <sourceCategoryApiName>pipeline</sourceCategoryApiName>
            <weight>1.0</weight>
        </weightedSourceCategories>
    </forecastingCategoryMappings>
```

Wildcard Support in the Manifest File

The wildcard character `*` (asterisk) in the `package.xml` manifest file doesn't apply to metadata types for feature settings. The wildcard applies only when retrieving all settings, not for an individual setting. For details, see Settings. For information about using the manifest file, see Deploying and Retrieving Metadata with the Zip File.

HighVelocitySalesSettings

Represents an org's High Velocity Sales settings. With High Velocity Sales, you can make your inside sales team as effective as possible. This type extends the Metadata metadata type and inherits its `fullName` field.

In the package manifest, all organization settings metadata types are accessed using the "Settings" name. See Settings for more details.

File Suffix and Directory Location

HighVelocitySalesSettings values are stored in a single file named `HighVelocitySales.settings` in the `settings` directory of the corresponding package directory.

Version

HighVelocitySalesSettings components are available in API version 47.0 and later.

Fields

Field Name	Field Type	Description
`enableACAutoSendEmail`	boolean	Indicates whether a sales cadence step of type Automated Send can be created `(true)` or not `(false)`. Allowing Salesforce to automatically send an email to a prospect can make your sales team more efficient, because reps do not have to send the email themselves. Default value is `(true)`. Available in API version 48.0 and later.
`enableDispositionCategory`	boolean	Indicates whether Call Outcomes For Branching is enabled in High Velocity Sales `(true)` or not `(false)`. Use Call Outcomes For Branching to group calls into different outcome categories such as "Left Voicemail" or "Not Interested." You can see the outcomes in a report, or use them to determine how sales cadences are branched. `enableHighVelocitySales` must be `(true)` to use High Velocity Sales. Default value is `false`.
`enableEngagementWaveAnalyticsPref`	boolean	Indicates whether you can see engagement statistics in Einstein Analytics `(true)` or not `(false)`. Use Einstein Analytics to analyze information about calls, engagement, and how each sales rep moves through their cadence steps.
`enableHighVelocitySales`	boolean	Indicates whether High Velocity Sales is enabled for this org `(true)` or not `(false)`. This turns on the features required for the product and makes the app available to users. Default value is `false`.
`enableHighVelocitySalesSetup`	boolean	Indicates whether High Velocity Sales is enabled `(true)` or not `(false)`. Default value is `false`.

Declarative Metadata Sample Definition

The following is an example of the `HighVelocitySales.settings` file:

```
<?xml version="1.0" encoding="UTF-8"?>
<HighVelocitySalesSettings xmlns="http://soap.sforce.com/2006/04/metadata">
 <enableACAutoSendEmail>false</enableACAutoSendEmail>
 <enableDispositionCategory>true</enableDispositionCategory>
 <enableEngagementWaveAnalyticsPref>true</enableEngagementWaveAnalyticsPref>
 <enableHighVelocitySales>true</enableHighVelocitySales>
 <enableHighVelocitySalesSetup>true</enableHighVelocitySalesSetup>
</HighVelocitySalesSettings>
```

Example Package Manifest

The following is an example package manifest used to deploy or retrieve the HighVelocitySalesSettings settings metadata:

```
<?xml version="1.0" encoding="UTF-8"?>
<Package xmlns="http://soap.sforce.com/2006/04/metadata">
 <types>
  <members>HighVelocitySalesSettings</members>
  <name>Settings</name>
 </types>
 <version>47.0</version>
</Package>
```

IdeasSettings

Represents the metadata used to manage settings for Ideas.

In the package manifest, all organization settings metadata types are accessed using the “Settings” name. See Settings for more details.

File Suffix and Directory Location

IdeasSettings is stored in one file named `Ideas.settings` in the `settings` folder of the corresponding package directory. The `.settings` files are different from other named components because there is only one settings file for each settings component.

Version

IdeasSettings is available in API version 27.0 and later.

Ideas

Represents settings for Ideas and Idea Themes.

Fields

Field Name	Field Type	Description
`enableIdeaThemes`	boolean	Indicates whether Idea Themes is enabled (`true`) or not (`false`).

Field Name	Field Type	Description
enableIdeas	boolean	Indicates whether Ideas is enabled (true) or not (false).
enableIdeasReputation	boolean	Indicates whether Reputation is enabled (true) or not (false). You can't enable IdeasReputation without enabling the Ideas Reputation permission in your organization. This field is available in API version 28.0 and later.
enableChatterProfile	boolean	Indicates that the Chatter user profile is used for Ideas user profiles. If enableChatterProfile is true, the ideasProfilePage value must not be specified. If enableChatterProfile is false, then specify a ideasProfilePage value, otherwise the Ideas zone profile is used. This field is available in API version 29.0 and later.
ideasProfilePage	string	The name of the Visualforce page to use for a custom Ideas user profile, if enableChatterProfile is false. If enableChatterProfile is false, then specify a ideasProfilePage value, otherwise the Ideas zone profile is used. This field is available in API version 29.0 and later.
halfLife	double	Indicates how quickly old ideas drop in ranking on the Popular Ideas subtab. The half-life setting determines how the number of days after which old ideas drop in ranking on the Popular Ideas subtab, to make room for ideas with more recent votes. A shorter half-life moves older ideas down the page faster than a longer half-life.

Declarative Metadata Sample Definition

The following is an example ideas.settings metadata file:

```
<?xml version="1.0" encoding="UTF-8"?>
<IdeasSettings xmlns="http://soap.sforce.com/2006/04/metadata">
   <enableIdeaThemes>true</enableIdeaThemes>
   <enableIdeas>true</enableIdeas>
   <enableIdeasReputation>true</enableIdeasReputation>
   <enableChatterProfile>false</enableChatterProfile>
   <ideasProfilePage>name of Visualforce page</ideasProfilePage>
   <halfLife>2.6</halfLife>
</IdeasSettings>
```

Wildcard Support in the Manifest File

The wildcard character * (asterisk) in the package.xml manifest file doesn't apply to metadata types for feature settings. The wildcard applies only when retrieving all settings, not for an individual setting. For details, see Settings. For information about using the manifest file, see Deploying and Retrieving Metadata with the Zip File.

IndustriesSettings

Represents settings for industries verticals like Financial Service Cloud, Consumer Goods Cloud, and Health Cloud.This type extends the Metadata metadata type and inherits its fullName field.

In the package manifest, all organization settings metadata types are accessed using the “Settings” name. See Settings for more details.

File Suffix and Directory Location

IndustriesSettings are stored in a single file named `Industries.settings` in the `settings` directory.

Version

Industries settings are available in API version 47.0 and later.

Special Access Rules

Some settings are specific to an industry vertical and are only available to customers with org editions where the vertical is enabled. Users need the FSC Insurance permission set to access the settings available in editions with Financial Services Cloud.

Fields for Financial Services Cloud

Field Name	Field Type	Description
`allowMultipleProducersToWorkOnSamePolicy`	boolean	Indicates whether multiple producers can be assigned to the same insurance policy. The default is `false`. This field is available in editions where Financial Services Cloud is enabled.
`enableAccessToMasterListOfCoverageTypes`	boolean	Indicates whether insurance agents can access the master list of coverage types. The default is `false`. This field is available in editions where Financial Services Cloud is enabled.
`enableBlockResourceAvailabilityOrgPref`	boolean	Indicates whether Lightning Scheduler service appointments are added to users' Salesforce calendars. For example, if set to `false`, users won’t see their service appointments on their calendars. The default is `false`.
`enableEventManagementOrgPref`	boolean	Indicates whether users can add multiple service resources to a service appointment. The default is `false`.
`enableFSCInsuranceReport`	boolean	Indicates whether sales managers can access the dashboard and prebuilt reports. The default is `false`. This field is available in API version 48.0 and later in editions where Financial Services Cloud is enabled. Note: This setting can be enabled only if the `allowMultipleProducersToWorkOnSamePolicy` setting is already set to `true`.
`enableManyToManyRelationships`	boolean	Indicates whether insurance can manage many-to-many relationships between claims and cases, claims and assets, and assets and policy participants. For example, if set to `true`, agents can handle multiple claims through one case or have multiple cases handling one claim. The default is `false`. This field is available in editions where Financial Services Cloud is enabled.

Field Name	Field Type	Description
enableMortgageRlaTotalsOrgPref	boolean	Indicates whether the calculation of assets and liabilities for residential loan application records is enabled for your org (true or not (false). The default is `false`. This field is available in editions where Financial Services Cloud is enabled.
enableMultiResourceOrgPref	boolean	Indicates whether users can add multiple service resources to a service appointment. The default is `false`.
enableOverbookingOrgPref	boolean	Indicates whether users can add multiple service appointments to a single time slot for a service resource. If set to `false`, concurrent time slots are visible, but can't be modified. The default is `false`.
enableReferralScoring	boolean	Indicates whether Einstein Referral Scoring for Financial Services Cloud is enabled for your org (true) or not (false). The default is `false`. This field is available in editions where Financial Services Cloud is enabled.

Fields for Health Cloud

Field Name	Field Type	Description
enableProviderSearchSyncOrgPref	boolean	Indicates whether provider data search is synced every six hours. The default is `false`. This field is available in editions where Health Cloud is enabled.

Fields for Consumer Goods Cloud

Field Name	Field Type	Description
enableObjectDetection	boolean	Indicates whether Einstein Object Detection is enabled for your org. The default is `false`. This field is available in editions where Consumer Goods Cloud is enabled.

Declarative Metadata Sample Definition

The following is an example of a Industries.Settings metadata file.

```
<?xml version="1.0" encoding="UTF-8"?>
<IndustriesSettings xmlns="http://soap.sforce.com/2006/04/metadata">
    <enableEventManagementOrgPref>true</enableEventManagementOrgPref>
    <enableMultiResourceOrgPref>false</enableMultiResourceOrgPref>
    <enableOverbookingOrgPref>true</enableOverbookingOrgPref>
    <enableBlockResourceAvailabilityOrgPref>true</enableBlockResourceAvailabilityOrgPref>

   <enableAccessToMasterListOfCoverageTypes>true</enableAccessToMasterListOfCoverageTypes>
```

```
    <enableManyToManyRelationships>true</enableManyToManyRelationships>

<allowMultipleProducersToWorkOnSamePolicy>false</allowMultipleProducersToWorkOnSamePolicy>

    <enableMortgageRlaTotalsOrgPref>true</enableMortgageRlaTotalsOrgPref>
    <enableObjectDetection>true</enableObjectDetection>
    <enableProviderSearchSyncOrgPref>true</enableProviderSearchSyncOrgPref>
    <enableReferralScoring>true</enableReferralScoring>
    <enableFSCInsuranceReport>true</enableFSCInsuranceReport>
</IndustriesSettings>
```

The following is an example `package.xml` that references the previous definition.

```
<?xml version="1.0" encoding="UTF-8"?>
<Package xmlns="http://soap.sforce.com/2006/04/metadata">
    <types>
        <members>Industries</members>
        <name>Settings</name>
    </types>
    <version>47.0</version>
</Package>
```

InvocableActionSettings

Represents the org's invocable action settings, such as whether partial save is allowed.This type extends the Metadata metadata type and inherits its `fullName` field.

File Suffix and Directory Location

InvocableActionSettings values are stored in the `InvocableAction.settings` file in the settings directory. The `.settings` files are different from other named components because there is only one settings file for each settings component.

Version

InvocableActionSettings components are available in API version 47.0 and later.

Fields

Field Name	Field Type	Description
`isPartialSaveAllowed`	boolean	Indicates whether partial save is enabled for most invocable actions that are invoked via REST API and executed in bulk. When the value is `true`, Salesforce tries three times to execute invocable actions that run successfully and rolls back only the invocable actions that fail to execute. This functionality is called partial save. If the field is set to `false`, if one invocable action fails, Salesforce rolls back other invocable actions in the same transaction and the entire transaction fails. Corresponds to the Enable Partial Save for Invocable Actions critical update.

Declarative Metadata Sample Definition

The following is an example of the InvocableAction.settings file.

```
<?xml version="1.0" encoding="UTF-8"?>
<InvocableActionSettings xmlns="http://soap.sforce.com/2006/04/metadata">
    <isPartialSaveAllowed>false</isPartialSaveAllowed>
</InvocableActionSettings>
```

Example Package Manifest

The following is an example package.xml that references the previous definition.

```
<?xml version="1.0" encoding="UTF-8"?>
<Package xmlns="http://soap.sforce.com/2006/04/metadata">
 <types>
  <members>InvocableAction</members>
  <name>Settings</name>
 </types>
 <version>47.0</version>
</Package>
```

Wildcard Support in the Manifest File

The wildcard character * (asterisk) in the package.xml manifest file doesn't apply to metadata types for feature settings. The wildcard applies only when retrieving all settings, not for an individual setting. For details, see Settings. For information about using the manifest file, see Deploying and Retrieving Metadata with the Zip File.

IoTSettings

Represents the organization's IoT settings, such as whether or not IoT or IoT Insights is enabled.This type extends the Metadata metadata type and inherits its fullName field.

File Suffix and Directory Location

IoTSettings components have the suffix IoT.settings and are stored in the settings folder. The .settings files are different from other named components because there is only one settings file for each settings component. In the package manifest, all organization settings metadata types are accessed using the Settings name.

Version

IoTSettings components are available in API version 44.0 and later.

Fields

Field Name	Field Type	Description
enableIoT	boolean	Indicates whether IoT is enabled (true) or not (false).

Declarative Metadata Sample Definition

The following is an example of a IoTSettings component.

```
<?xml version="1.0" encoding="UTF-8"?>
<IoTSettings xmlns="http://soap.sforce.com/2006/04/metadata">
<enableIoT>true</enableIoT>
</IoTSettings>
```

Wildcard Support in the Manifest File

The wildcard character * (asterisk) in the `package.xml` manifest file doesn't apply to metadata types for feature settings. The wildcard applies only when retrieving all settings, not for an individual setting. For details, see Settings. For information about using the manifest file, see Deploying and Retrieving Metadata with the Zip File.

KnowledgeSettings

Represents the metadata used to manage settings for Salesforce Knowledge.

This type extends the Metadata metadata type and inherits its `fullName` field.

In the package manifest, all organization settings metadata types are accessed using the "Settings" name. See Settings for more details.

File Suffix and Directory Location

KnowledgeSettings values are stored in a single file named `Knowledge.settings` in the `settings` directory. The `.settings` files are different from other named components because there is only one settings file for each settings component.

Version

KnowledgeSettings is available in API version 27.0 and later.

Fields

Field Name	Field Type	Description
`answers`	KnowledgeAnswerSettings	Represents the metadata used to manage settings for Salesforce Knowledge and Answers.
`cases`	KnowledgeCaseSettings	Represents the metadata used to manage settings for Salesforce Knowledge and Cases.
`defaultLanguage`	string	Required. The default language for Salesforce Knowledge. Use the abbreviation for the language, for example, en_US for United States English.
`enableChatterQuestionKBDeflection`	boolean	Indicates whether tracking for case deflection via Chatter is enabled (`true`) or not (`false`).

Field Name	Field Type	Description
enableCreateEditOnArticlesTab	boolean	Indicates whether users can create and edit articles on the articles tab (true) or not (false).
enableExternalMediaContent	boolean	Indicates whether connecting to external media is enabled (true) or not (false).
enableKnowledge	boolean	Indicates whether Salesforce Knowledge is enabled (true) or not (false). This field is false by default.
enableKnowledgeArticleTextHighlights	boolean	Indicates whether text snippet highlights in Salesforce Knowledge search results are enabled (true) or not (false). This field is true by default. Available in API version 47.0 and later.
enableKnowledgeKeywordAutoComplete	boolean	Indicates whether auto-complete for keywords is enabled (true) or not (false) when searching Salesforce Knowledge. This field is true by default. Available in API version 47.0 and later.
enableKnowledgeTitleAutoComplete	boolean	Indicates whether auto-complete for article titles is enabled (true) or not (false) when searching Salesforce Knowledge. This field is true by default. Available in API version 47.0 and later.
enableLightningKbAutoLoadRichTextField	boolean	Indicates whether rich text fields are enabled for editing when an article loads in Lightning Knowledge (true) or not (false). This field is false by default. Available in API version 47.0 and later.
enableLightningKnowledge	boolean	Indicates whether Lightning Knowledge is enabled (true) or not (false).
languages	KnowledgeLanguageSettings	A list of languages enabled for Salesforce Knowledge.
showArticleSummariesCustomerPortal	boolean	Indicates whether article summaries appear in the Customer Portal (true) or not (false).
showArticleSummariesInternalApp	boolean	Indicates whether article summaries appear in the internal knowledge base (true) or not (false).
showArticleSummariesPartnerPortal	boolean	Indicates whether article summaries appear in the partner portal (true) or not (false).
showValidationStatusField	boolean	Indicates whether validation status appears on articles (true) or not (false).

Field Name	Field Type	Description
suggestedArticles	KnowledgeSuggestedArticlesSettings	Represents the metadata used to manage settings for the case fields used to suggest articles for cases. Available in API version 37.0 and later.

KnowledgeAnswerSettings

Represents the metadata used to manage settings for Salesforce Knowledge and Answers.

Field Name	Field Type	Description
assignTo	string	Specifies the username an article is assigned to from Answers.
defaultArticleType	string	The default article type for articles created from Answers. Uses the API name of the article type.
enableArticleCreation	boolean	Indicates whether users can create articles from Answers (true) or not (false).

KnowledgeCaseSettings

Represents the metadata used to manage settings for Salesforce Knowledge and Cases.

Field Name	Field Type	Description
articlePDFCreationProfile	string	The profile used to create a PDF of an article from Cases.
articlePublicSharingSites	KnowledgeSitesSettings	Represents the metadata used to manage settings for Salesforce Knowledge and Sites.
articlePublicSharingCommunities	KnowledgeSitesSettings	Represents the metadata used to manage settings for Salesforce Knowledge and Communities.
articlePublicSharingSitesChatterAnswers	KnowledgeSitesSettings	Represents the metadata used to manage settings for Salesforce Knowledge and Sites with Chatter Answers.
assignTo	string	Specifies the username an article is assigned to from Cases.
customizationClass	string	Specifies the Apex class used for customization.
defaultContributionArticleType	string	The default article type for articles created from Cases.
editor	KnowledgeCaseEditor (enumeration of type string)	Indicates the rich text editor type. Valid values are: • simple • standard

Field Name	Field Type	Description
`enableArticleCreation`	boolean	Indicates whether users can create articles from Cases (`true`) or not (`false`). Controls whether other fields on KnowledgeCaseSettings can be set.
`enableArticlePublicSharingSites`	boolean	Indicates whether articles can be shared via a public site (URL) from Cases (`true`) or not (`false`).
`enableCaseDataCategoryMapping`	boolean	Indicates whether Case Data Category mapping is enabled (`true`) or not (`false`).
`useProfileForPDFCreation`	boolean	Indicates whether a profile is used to create a PDF of an article from Cases (`true`) or not (`false`).

KnowledgeSitesSettings

Represents the metadata used to manage settings for Salesforce Knowledge and Sites.

Field Name	Field Type	Description
`site`	string[]	Specifies the site used for Salesforce Knowledge and Sites.

KnowledgeLanguageSettings

A list of languages enabled for Salesforce Knowledge. KnowledgeLanguageSettings is available in API version 28.0 and later.

Field Name	Field Type	Description
`language`	KnowledgeLanguage	Represents the metadata used to manage settings for the languages enabled for Salesforce Knowledge.

KnowledgeLanguage

Represents the metadata used to manage settings for the languages enabled for Salesforce Knowledge. KnowledgeLanguage is available in API version 28.0 and later.

Field Name	Field Type	Description
`active`	boolean	Indicates whether the language is enabled (`true`) or not (`false`).
`defaultAssignee`	string	The default assignee for articles in the language.
`defaultAssigneeType`	KnowledgeLanguageLookupValueType (enumeration of type string)	Indicates the default assignee type. Valid values are: • `User` • `Queue`
`defaultReviewer`	string	The default reviewer for articles in the language.

Field Name	Field Type	Description
`defaultReviewerType`	KnowledgeLanguageLookupValueType (enumeration of type string)	Indicates the default reviewer type. Valid values are: - `User` - `Queue`
`name`	string	The code for the language name, for example: English is `en`. See "What languages does Salesforce support?" in the Salesforce Help for a list of supported languages and their codes.

KnowledgeSuggestedArticlesSettings

Represents the metadata used to manage settings for the articles suggested for cases, work orders, and work order line items. The Work Order and Work Order Line Item objects must be enabled in the org to use the associated fields.

Field Name	Field Type	Description
`caseFields`	KnowledgeCaseFieldsSettings	Represents a list of the case fields used to suggest articles for the case.
`useSuggestedArticlesForCase`	boolean	Indicates whether case content is used to suggest articles for cases (true) or not (false).
`workOrderFields`	KnowledgeWorkOrderFieldsSettings	Represents a list of the work order fields used to suggest articles for the work order.
`workOrderLineItemFields`	KnowledgeWorkOrderLineItemFieldsSettings	Represents a list of the work order line item fields used to suggest articles for the work order line item.

KnowledgeCaseFieldsSettings

Represents a list of the case fields used to suggest articles for the case. Available in API version 37.0 and later.

Field Name	Field Type	Description
`field`	KnowledgeCaseField[]	Specifies the names of the case fields used to suggest articles for the case.

KnowledgeCaseField

Represents the name of the case field used to suggest articles for the case. Available in API version 37.0 and later.

Field Name	Field Type	Description
`name`	string	Specifies the name of the case field used to suggest articles for the case.

KnowledgeWorkOrderFieldsSettings

Represents a list of the work order fields used to suggest articles for the work order. Available in API version 39.0 and later.

Field Name	Field Type	Description
`field`	KnowledgeWorkOrderField[]	Specifies the names of the work order fields used to suggest articles for the work order.

KnowledgeWorkOrderField

Represents the name of the work order field used to suggest articles for the work order. Available in API version 39.0 and later.

Field Name	Field Type	Description
`name`	string	Specifies the name of the work order field used to suggest articles for the work order.

KnowledgeWorkOrderLineItemFieldsSettings

Represents a list of the work order line item fields used to suggest articles for the work order line item. Available in API version 39.0 and later.

Field Name	Field Type	Description
`field`	KnowledgeWorkOrderLineItemField[]	Specifies the names of the work order line item fields used to suggest articles for the work order line item.

KnowledgeWorkOrderLineItemField

Represents the name of the work order line item field used to suggest articles for the work order line item. Available in API version 39.0 and later.

Field Name	Field Type	Description
`name`	string	Specifies the name of the work order line item field used to suggest articles for the work order line item.

Declarative Metadata Sample Definition

This is a sample Knowledge settings file.

```xml
<?xml version="1.0" encoding="UTF-8"?>
<KnowledgeSettings xmlns="http://soap.sforce.com/2006/04/metadata">
    <answers>
        <enableArticleCreation>false</enableArticleCreation>
    </answers>
    <cases>
        <articlePDFCreationProfile>partner portal knowledge
```

```
profile</articlePDFCreationProfile>
        <articlePublicSharingSites>
            <site>KnowledgeSite</site>
            <site>PKB2Site</site>
            <site>ChatterAnswersSite</site>
        </articlePublicSharingSites>
        <articlePublicSharingSitesChatterAnswers>
            <site>ChatterAnswersSite</site>
        </articlePublicSharingSitesChatterAnswers>
        <assignTo>testall@kb.org</assignTo>
        <defaultContributionArticleType>Support</defaultContributionArticleType>
        <editor>simple</editor>
        <enableArticleCreation>true</enableArticleCreation>
        <enableArticlePublicSharingSites>true</enableArticlePublicSharingSites>
        <useProfileForPDFCreation>true</useProfileForPDFCreation>
    </cases>
    <defaultLanguage>ja</defaultLanguage>
    <enableCreateEditOnArticlesTab>true</enableCreateEditOnArticlesTab>
    <enableExternalMediaContent>true</enableExternalMediaContent>
    <enableKnowledge>true</enableKnowledge>
    <showArticleSummariesCustomerPortal>true</showArticleSummariesCustomerPortal>
    <showArticleSummariesInternalApp>true</showArticleSummariesInternalApp>
    <showArticleSummariesPartnerPortal>true</showArticleSummariesPartnerPortal>
    <showValidationStatusField>true</showValidationStatusField>
    <suggestedArticles>
      <caseFields>
         <field>
            <name>Subject</name>
         </field>
         <field>
            <name>SuppliedEmail</name>
         </field>
      </caseFields>
      <useSuggestedArticlesForCase>true</useSuggestedArticlesForCase>
    </suggestedArticles>
</KnowledgeSettings>
```

Wildcard Support in the Manifest File

The wildcard character * (asterisk) in the package.xml manifest file doesn't apply to metadata types for feature settings. The wildcard applies only when retrieving all settings, not for an individual setting. For details, see Settings. For information about using the manifest file, see Deploying and Retrieving Metadata with the Zip File.

LanguageSettings

Represents an organization's language settings. Language settings control end-user language selection, locale formats, and translation options. This type extends the Metadata metadata type and inherits its fullName field.

In the package manifest, all organization settings metadata types are accessed using the "Settings" name. See Settings for more details.

File Suffix and Directory Location

LanguageSettings values are stored in the `Language.settings` file in the `settings` directory. The `.settings` files are different from other named components because there is only one settings file for each settings component.

Version

LanguageSettings is available in API version 47.0 and later.

Fields

Field Name	Field Type	Description
`enableCanadaIcuFormat`	boolean	Indicates whether the ICU format is enabled for the en_CA locale (`true`) or not (`false`). This field has a default value of `true` for orgs created in API version 47.0 and later. Orgs created prior to API version 47.0 have a default of `false`.
`enableEndUserLanguages`	boolean	Indicates whether end-user languages are enabled (`true`) or not (`false`). This field has a default value of `false`.
`enableICULocaleDateFormat`	boolean	Indicates whether date and currency are formatted with the International Components for Unicode (`true`) or not (`false`). This field has a default value of `false`. See Go Global with New International Locale Formats for more information.
`enableLocalNamesForStdObjects`	boolean	Indicates whether local name fields can be defined for standard objects (`true`) or not (`false`). This has a default value of `false`. This field is available in API version 48.0 and later.
`enablePlatformLanguages`	boolean	Indicates whether platform-only languages are enabled (`true`) or not (`false`). This field has a default value of `false`. Setting this field to `true` also sets `enableEndUserLanguages` to `true`.
`enableTranslationWorkbench`	boolean	Indicates whether the Translation Workbench is enabled (`true`) or not (`false`). This field has a default value of `false`.
`useLanguageFallback`	boolean	Indicates whether translation follows the language fallback rule (`true`) or returns the master label (`false`). This field has a default value of `true`.

Declarative Metadata Sample Definition

The following is an example of a LanguageSettings file.

```xml
<?xml version="1.0" encoding="UTF-8"?>
<LanguageSettings xmlns="http://soap.sforce.com/2006/04/metadata">
    <enableCanadaIcuFormat>true</enableCanadaIcuFormat>
    <enableEndUserLanguages>true</enableEndUserLanguages>
    <enableICULocaleDateFormat>true</enableICULocaleDateFormat>
    <enableLocalNamesForStdObjects>false</enableLocalNamesForStdObjects>
    <enablePlatformLanguages>false</enablePlatformLanguages>
    <enableTranslationWorkbench>true</enableTranslationWorkbench>
    <useLanguageFallback>true</useLanguageFallback>
</LanguageSettings>
```

The following is an example `package.xml` that references the previous definition.

```
<?xml version="1.0" encoding="UTF-8"?>
<Package xmlns="http://soap.sforce.com/2006/04/metadata">
    <types>
        <members>Language</members>
        <name>Settings</name>
    </types>
    <version>48.0</version>
</Package>
```

Wildcard Support in the Manifest File

The wildcard character `*` (asterisk) in the `package.xml` manifest file doesn't apply to metadata types for feature settings. The wildcard applies only when retrieving all settings, not for an individual setting. For details, see Settings. For information about using the manifest file, see Deploying and Retrieving Metadata with the Zip File.

LeadConfigSettings

Represents configuration settings for Leads that control how they are converted and displayed, and what actions are available. This type extends the Metadata metadata type and inherits its `fullName` field.

In the package manifest, all organization settings metadata types are accessed using the "Settings" name. See Settings for more details.

File Suffix and Directory Location

LeadConfigSettings values are stored in the `LeadConfig.settings` file in the `settings` folder. The `.settings` files are different from other named components because there is only one settings file for each settings component.

Version

LeadConfigSettings is available in API versions 47.0 and later.

Fields

Field Name	Field Type	Description
doesEnableLeadConvertDefaultSubjectBlankTaskCreation	boolean	Configures whether tasks without a subject are created during lead conversion. If `true`, tasks are created when the default subject field has no value. If `false`, only tasks with a subject are created.
doesHideOpportunityInConvertLeadWindow	boolean	Hides the opportunity section of the Convert Lead window during the conversion of a lead. Default value is `false`.

Field Name	Field Type	Description
doesPreserveLeadStatus	boolean	If your organization uses record types, the lead status changes to the lead status value of the new owner's record type during conversion. Set doesPreserveLeadStatus to true to preserve the value of the lead status during conversion. Orgs that use record types can create a lead process that allows different lead status values for different record types. If doesPreserveLeadStatus is false, the lead status might change during lead conversion if the new owner's record type has a different default value for lead status. Default value is true.
doesSelectNoOpportunityOnConvertLead	boolean	Prevents an opportunity from being created when the lead is converted. Default value is false.
doesTrackHistory	boolean	Enables field history tracking for leads. When field history tracking is enabled, users can choose the fields they want to track. Default value is false.
enableConversionsOnMobile	boolean	Lets a user convert leads on their mobile devices. The Convert Lead action converts qualified leads to contacts. Default value is true.
enableOrgWideMergeAndDelete	boolean	Lets a user merge and delete leads. The user must also have the Public Read/Write/Transfer permission. Default value is false.
shouldLeadConvertRequireValidation	boolean	Enforces validation rules when converting leads. Default value is true.

Declarative Metadata Sample Definition

The following is an example of the LeadConfigSettings type:

```
<?xml version="1.0" encoding="UTF-8"?>
<LeadConfigSettings xmlns="http://soap.sforce.com/2006/04/metadata">


<doesEnableLeadConvertDefaultSubjectBlankTaskCreation>true</doesEnableLeadConvertDefaultSubjectBlankTaskCreation>

    <enableConversionsOnMobile>true</enableConversionsOnMobile>
    <shouldLeadConvertRequireValidation>true</shouldLeadConvertRequireValidation>
    <enableOrgWideMergeAndDelete>false</enableOrgWideMergeAndDelete>
    <doesPreserveLeadStatus>true</doesPreserveLeadStatus>
    <doesSelectNoOpportunityOnConvertLead>false</doesSelectNoOpportunityOnConvertLead>
   <doesHideOpportunityInConvertLeadWindow>false</doesHideOpportunityInConvertLeadWindow>
```

```
    <doesTrackHistory>false</doesTrackHistory>
</LeadConfigSettings>
```

Example Package Manifest

The following is an example package manifest used to deploy or retrieve the Account settings metadata for an organization:

```
<?xml version="1.0" encoding="UTF-8"?>
<!--

<Package xmlns="http://soap.sforce.com/2006/04/metadata">
    <types>
        <members>LeadConfig</members>
        <name>Settings</name>
    </types>
    <version>47.0</version>
</Package>
```

LeadConvertSettings

Represents an organization's custom field mappings for lead conversion. Custom fields can be mapped from Leads to Accounts, Contacts, and Opportunities. Options for creating opportunities during lead conversion can also be specified. This type extends the Metadata metadata type and inherits its `fullName` field.

Version

LeadConvertSettings is available in API versions 39.0 and later.

Fields

Field Name	Field Type	Description
`allowOwnerChange`	boolean	Indicates whether to include the RecordOwner field in the Convert Lead dialog box (`true`) or not (`false`).
`objectMapping`	metadata type	A set of `inputObject`, `mappingFields`, and `outputObject` entries. Up to three `objectMapping` types can be declared—one each for Account, Contact, and Opportunity.
`inputObject`	string	The name of the object type containing the source fields for mapping. The value will always be `Lead`.
`mappingFields`	metadata type	A set of `inputField` and `outputField` entries.
`inputField`	string	The name of a custom lead field supplying source data during lead conversion.
`outputField`	string	The name of a custom account, contact, or opportunity field that will receive data from source field named in the accompanying `inputField` entry.

Field Name	Field Type	Description
`outputObject`	string	The name of the object type receiving data during lead conversion—`Account`, `Contact`, or `Opportunity`.
`opportunityCreationOptions`	string	This optional field determines whether the Opportunity field is visible or required in the Convert Lead dialog box. Valid values include: • `VisibleOptional`—The Opportunity field is included in the dialog box but not required. A new opportunity is created if the user enters an opportunity name. This is the default value. • `VisibleRequired`—The Opportunity field is included in the dialog box and is required. A new opportunity is created based on the name entered by the user. • `NotVisible`—The Opportunity field is not included in the dialog box. No opportunity is created.

Declarative Metadata Sample Definition

The following is an example of the LeadConvertSettings type:

```xml
<?xml version="1.0" encoding="UTF-8"?>
<LeadConvertSettings xmlns="http://soap.sforce.com/2006/04/metadata">
    <allowOwnerChange>false</allowOwnerChange>
    <objectMapping>
        <inputObject>Lead</inputObject>
        <mappingFields>
            <inputField>custom_lead_field_1</inputField>
            <outputField>custom_account_field_1</outputField>
        </mappingFields>
        <mappingFields>
            <inputField>custom_lead_field_2</inputField>
            <outputField>custom_account_field_2</outputField>
        </mappingFields>
        <mappingFields>
            <inputField>custom_lead_field_3</inputField>
            <outputField>custom_account_field_3</outputField>
        </mappingFields>
        <outputObject>Account</outputObject>
    </objectMapping>
    <objectMapping>
        <inputObject>Lead</inputObject>
        <mappingFields>
            <inputField>custom_lead_field_4</inputField>
            <outputField>custom_opportunity_field_1</outputField>
        </mappingFields>
        <outputObject>Opportunity</outputObject>
    </objectMapping>
    <opportunityCreationOptions>VisibleOptional</opportunityCreationOptions>
</LeadConvertSettings>
```

Wildcard Support in the Manifest File

The wildcard character * (asterisk) in the package.xml manifest file doesn't apply to metadata types for feature settings. The wildcard applies only when retrieving all settings, not for an individual setting. For details, see Settings. For information about using the manifest file, see Deploying and Retrieving Metadata with the Zip File.

LiveAgentSettings

Represents an organization's Chat settings, such as whether or not Chat is enabled. This type extends the Metadata metadata type and inherits its fullName field.

File Suffix and Directory Location

LiveAgentSettings values are stored in the LiveAgent.settings file in the settings directory. The .settings files are different from other named components because there is only one settings file for each settings component.

In the package manifest, all organization settings metadata types are accessed using the "Settings" name. See Settings for more details.

Version

LiveAgentSettings is available in API version 28.0 and later.

Fields

Field Name	Field Type	Description
enableLiveAgent	boolean	Indicates whether Chat is enabled (true) or not (false).
enableQuickTextEnabled	boolean	Indicates whether Quick Text is enabled (true) or not (false).
priority	integer	Indicates the priority level of a Chat.

Declarative Metadata Sample Definition

This is a sample Chat settings file.

```
<?xml version="1.0" encoding="UTF-8"?>
<LiveAgentSettings xmlns="http://soap.sforce.com/2006/04/metadata">
    <enableLiveAgent>true</enableLiveAgent>
</LiveAgentSettings>
```

Wildcard Support in the Manifest File

The wildcard character * (asterisk) in the package.xml manifest file doesn't apply to metadata types for feature settings. The wildcard applies only when retrieving all settings, not for an individual setting. For details, see Settings. For information about using the manifest file, see Deploying and Retrieving Metadata with the Zip File.

LightningExperienceSettings

Represents the settings that modify an org's Lightning Experience configuration. This type extends the Metadata metadata type and inherits its `fullName` field.

In the package manifest, all organization settings metadata types are accessed using the "Settings" name. See Settings for more details.

File Suffix and Directory Location

A LightningExperienceSettings component has the suffix `.settings` and is stored in the `settings` folder. The `.settings` files are different from other named components because there is only one settings file for each settings component.

Version

LightningExperienceSettings components are available in API version 47.0 and later.

Fields

Field Name	Field Type	Description
`activeThemeName`	string	Specifies the name of the theme that is currently active in the org. This field is available in API version 48.0 and later.
`enableAccessCheckCrucPref`	boolean	Deprecated in API version 47.0 and later because the feature is no longer available.
`enableApiUserLtngOutAccessPref`	boolean	Deprecated in API version 48.0 and later because the feature is no longer available. This field corresponds to the API Only Users Can Access Only Salesforce APIs critical update, which was enforced in Spring '20. If a user has the API Only User permission, they can access Salesforce only via APIs, regardless of their other permissions. This restriction already applied to other Salesforce features, but the critical update enforced the restriction in Lightning Out also.
`enableAuraCDNPref`	boolean	Indicates whether Lightning Experience and other apps use a content delivery network (CDN) to serve the static content for Lightning Component framework. A CDN generally speeds up page load time, but it also changes the source domain that serves the files. If your company has IP range restrictions for content served from Salesforce, test thoroughly before enabling this setting. The default is `true`.
`enableAuraDepAccessChksCRUCPref`	boolean	Indicates whether the Enable Dependency Access Checks In Lightning Components critical update is enforced (`true`) or not (`false`). This critical update improves Lightning component access checks by checking the access level of all component dependencies. This field is available in API version 48.0 and later.
`enableFeedbackInMobile`	boolean	Indicates whether users can send feedback to Salesforce from the mobile app. The default is `false`.
`enableGoogleSheetsForSfdcEnabled`	boolean	Reserved. Do not use.

Field Name	Field Type	Description
`enableIE11DeprecationMsgHidden`	boolean	Deprecated in API version 47.0 and later because the feature is no longer available.
`enableIE11LEXCrucPref`	boolean	Deprecated in API version 47.0 and later because the feature is no longer available.
`enableInAppTooltips`	boolean	Indicates whether users see onboarding tips in the mobile app. The default is `false`.
`enableLEXOnIpadEnabled`	boolean	Indicates whether Lightning Experience is turned on for iPad Browsers (`true`) or not (`false`). The default is `false`. See Give Users Access to Lightning Experience on iPad Browsers (Beta) in Salesforce Help.
`enableLexEndUsersNoSwitching`	boolean	Indicates whether Salesforce Classic is turned off for your org (`true`) or not (`false`). Removes the Switcher for all users in the org. The default is `false`. See Turn Off Salesforce Classic for Your Org in Salesforce Help. This field is similar to `enableUsersAreLightningOnly`. If either field is set to `true`, users are blocked from switching to Salesforce Classic.
`enableNavPersonalizationOptOut`	boolean	Indicates whether users are blocked from personalizing the Lightning Experience navigation bar (`true`) or not (`false`). The default is `false` (that is, users can personalize the navigation bar by default). Salesforce recommends disabling personalization at the app level, not the org level. See Configure User Interface Settings in Salesforce Help.
`enableRemoveThemeBrandBanner`	boolean	Deprecated in API version 47.0 and later because the feature is no longer available.
`enableS1BannerPref`	boolean	Deprecated in API version 47.0 and later because the feature is no longer available.
`enableS1BrowserEnabled`	boolean	Indicates whether all users can access the Salesforce mobile web view from a supported mobile browser (`true`) or not (`false`). If `false`, then users must access the Salesforce mobile full site view from a mobile browser. Full site view doesn't have the full functionality of mobile web view. Salesforce Classic and Lightning Experience aren't supported on mobile browsers.
`enableSkypeChatEnabled`	boolean	Indicates that Skype for Salesforce is turned on in the org (`true`) or not (`false`). See Chat with Skype for Salesforce. This field is available in API version 48.0 and later.
`enableS1DesktopEnabled`	boolean	Indicates whether Lightning Experience is turned on in the org (`true`) or not (`false`). After it is enabled, this setting can't be disabled via the user interface or the API. See Turn on Lightning Experience for Your Org
`enableS1UiLoggingEnabled`	boolean	Deprecated in API version 47.0 and later because the feature is no longer available.
`enableSparkAllUsers`	boolean	Indicates whether all users can access Webex Teams (`true`) or not (`false`). This field is valid only when

Field Name	Field Type	Description
		enableSparkConversationEnabled is set to true. See Give Users Cisco Webex Teams and Cisco Webex Meetings Permissions. This field is available in API version 48.0 and later.
enableSparkConversationEnabled	boolean	Indicates whether Webex Teams is turned on for your org (true) or not (false). This field is available in API version 48.0 and later. See Give Users Cisco Webex Teams and Cisco Webex Meetings Permissions.
enableTryLightningOptOut	boolean	Indicates whether the Try Lightning Experience Now prompt is hidden from users (true) or not (false). The default is false. See Try Lightning Experience Now Prompt in Salesforce Help.
enableUseS1AlohaDesktop	boolean	Deprecated in API version 47.0 and later because the feature is no longer available.
enableUsersAreLightningOnly	boolean	Indicates whether Salesforce Classic is turned off for your org (true) or not (false). Removes the Switcher for all users in the org. The default is false. This field is similar to enableLexEndUsersNoSwitching. If either field is set to true, users are blocked from switching to Salesforce Classic.
enableWebExEnabled	boolean	Indicates whether Webex Meetings is turned on for your org (true) or not (false). See Give Users Cisco Webex Teams and Cisco Webex Meetings Permissions. This field is available in API version 48.0 and later.
enableWebexAllUsers	boolean	Indicates whether all users can access Webex Meetings (true) or not (false). This field is valid only when enableWebExEnabled is set to true. See Give Users Cisco Webex Teams and Cisco Webex Meetings Permissions. This field is available in API version 48.0 and later.
isLEXExtensionComponentCustomizationOff	boolean	Indicates whether all users can enable the Lightning Extension Component Customization feature. If false, the feature is disabled for all users, even users who had it enabled. See Try New Features with the Lightning Extension for Chrome. This field is available in API version 48.0 and later.
isLEXExtensionDarkModeOff	boolean	Indicates whether all users can enable the Lightning Extension Dark Mode feature. If false, the feature is disabled for all users, even users who had it enabled. This field is available in API version 48.0 and later.
isLEXExtensionLinkGrabberOff	boolean	Indicates whether all users can enable the Lightning Extension Link Grabber feature. If false, the feature is disabled for all users, even users who had it enabled. This field is available in API version 48.0 and later.
isLEXExtensionOff	boolean	Indicates whether all users can enable the Lightning Extension for your org. If false, your users can't enable the Lightning Extension, even if they already have it installed. This field is available in API version 48.0 and later.

Declarative Metadata Sample Definition

The following is an example of a LightningExperienceSettings component.

```
<?xml version="1.0" encoding="UTF-8"?>
<LightningExperienceSettings xmlns="http://soap.sforce.com/2006/04/metadata">

    <enableAuraCDNPref>true</enableAuraCDNPref>
    <enableFeedbackInMobile>true</enableFeedbackInMobile>
    <enableInAppTooltips>true</enableInAppTooltips>
    <enableLEXOnIpadEnabled>true</enableLEXOnIpadEnabled>
    <enableLexEndUsersNoSwitching>true</enableLexEndUsersNoSwitching>
    <enableNavPersonalizationOptOut>true</enableNavPersonalizationOptOut>
    <enableS1BrowserEnabled>false</enableS1BrowserEnabled>
    <enableS1DesktopEnabled>true</enableS1DesktopEnabled>
    <enableSkypeChatEnabled>false</enableSkypeChatEnabled>
    <enableTryLightningOptOut>true</enableTryLightningOptOut>
    <enableUsersAreLightningOnly>true</enableUsersAreLightningOnly>
</LightningExperienceSettings>
```

The following is an example `package.xml` that references the previous definition.

```
<?xml version="1.0" encoding="UTF-8"?>
<Package xmlns="http://soap.sforce.com/2006/04/metadata">
    <types>
        <members>LightningExperience</members>
        <name>Settings</name>
    </types>
    <version>47.0</version>
</Package>
```

Wildcard Support in the Manifest File

The wildcard character `*` (asterisk) in the `package.xml` manifest file doesn't apply to metadata types for feature settings. The wildcard applies only when retrieving all settings, not for an individual setting. For details, see Settings. For information about using the manifest file, see Deploying and Retrieving Metadata with the Zip File.

LiveMessageSettings

Represents an org's LiveMessage settings.

Version

LiveMessageSettings components are available in API version 42.0 and later.

File Suffix and Directory Location

LiveMessageSettings values are stored in the LiveMessage.settings file in the settings folder. The .settings files are different from other named components because there is only one settings file for each settings component.

Fields

Field Name	Field Type	Description
enableLiveMessage	boolean	Turns LiveMessage on (true) or off (false) in an org. The default value is false.

Declarative Metadata Sample Definition

The following is an example of a liveMessageSettings component.

```
<?xml version="1.0" encoding="UTF-8"?>
<LiveMessageSettings xmlns="http://soap.sforce.com/2006/04/metadata">
   <enableLiveMessage>true</enableLiveMessage>
</LiveMessageSettings>
```

The following is an example package.xml that references the previous definition.

```
<?xml version="1.0" encoding="UTF-8"?>
<Package xmlns="http://soap.sforce.com/2006/04/metadata">
    <types>
        <members>LiveMessage</members>
        <name>Settings</name>
    </types>
    <version>44.0</version>
</Package>
```

Wildcard Support in the Manifest File

The wildcard character * (asterisk) in the package.xml manifest file doesn't apply to metadata types for feature settings. The wildcard applies only when retrieving all settings, not for an individual setting. For details, see Settings. For information about using the manifest file, see Deploying and Retrieving Metadata with the Zip File.

MacroSettings

Represents an organization's Macro settings, such as whether or not folders is enabled. This type extends the Metadata metadata type and inherits its fullName field.

File Suffix and Directory Location

MacroSettings values are stored in the Macro.settings file in the settings directory. The .settings files are different from other named components because there is only one settings file for each settings component.

In the package manifest, all organization settings metadata types are accessed using the "Settings" name. See Settings for more details.

Version

MacroSettings is available in API version 39.0 and later.

Fields

Field Name	Field Type	Description
enableAdvancedSearch	boolean	Indicates whether users can search all macro text fields (true) or not (false).
macrosInFolders	boolean	Indicates whether users can organize and share macros using folders (true) or not (false). Available in API version 44.0 and later.

Declarative Metadata Sample Definition

This is a sample Macro settings file.

```
<?xml version="1.0" encoding="UTF-8"?>
<MacroSettings xmlns="http://soap.sforce.com/2006/04/metadata">
    <enableAdvancedSearch>true</enableAdvancedSearch>
    <macrosInFolders>true</macrosInFolders>
</MacroSettings>
```

Wildcard Support in the Manifest File

The wildcard character * (asterisk) in the package.xml manifest file doesn't apply to metadata types for feature settings. The wildcard applies only when retrieving all settings, not for an individual setting. For details, see Settings. For information about using the manifest file, see Deploying and Retrieving Metadata with the Zip File.

MapAndLocationSettings

Represents an org's map and location settings.

In the package manifest, all organization settings metadata types are accessed using the "Settings" name. See Settings for more details.

Declarative Metadata File Suffix and Directory Location

MapAndLocationSettings values are stored in a single file named Mapandlocation.settings in the settings directory. The .settings files are different from other named components because there is only one settings file for each settings component.

Version

Map and location settings are available in API version 46.0 and later.

Fields

Field	Field Type	Description
enableAddressAutoComplete	boolean	Indicates whether auto-complete is enabled on address fields (true) or not (false).

Field	Field Type	Description
enableMapsAndLocation	boolean	Indicates whether the maps and location services are enabled (true) or not (false)

Declarative Metadata Sample Definition

This is a sample mapandlocation.settings metadata file.

```
<?xml version="1.0" encoding="UTF-8"?>
<MapsAndLocationSettings xmlns="http://soap.sforce.com/2006/04/metadata">
    <enableAddressAutoComplete>false</enableAddressAutoComplete>
    <enableMapsAndLocation>false</enableMapsAndLocation>
</MapsAndLocationSettings>
```

Wildcard Support in the Manifest File

The wildcard character * (asterisk) in the package.xml manifest file doesn't apply to metadata types for feature settings. The wildcard applies only when retrieving all settings, not for an individual setting. For details, see Settings. For information about using the manifest file, see Deploying and Retrieving Metadata with the Zip File.

MobileSettings

Represents an organization's mobile settings. This type extends the Metadata metadata type and inherits its fullName field.

In the package manifest, all organization settings metadata types are accessed using the "Settings" name. See Settings for more details.

Note: MobileSettings is no longer available in API versions 25.0 and 26.0.

Declarative Metadata File Suffix and Directory Location

MobileSettings values are stored in a single file named Mobile.settings in the settings directory. The .settings files are different from other named components because there is only one settings file for each settings component.

Version

Mobile settings are available in API version 27.0 and later.

Fields

Field	Field Type	Description
chatterMobile (Deprecated)	ChatterMobileSettings	Deprecated in API version 46.0. The settings for devices running Chatter mobile.
dashboardMobile (Deprecated)	DashboardMobileSettings	The settings for devices running the mobile dashboards app.

Field	Field Type	Description
`enableImportContactFromDevice`	boolean	Indicates whether users can import contacts from their mobile device (`true`) or not (`false`). Available in API version 47.0 and later.
`enableLightningOnMobile`	boolean	Removed in API version 48.0. Indicates whether the org is enabled for the new Salesforce mobile app. Available in API version 47.0 only.
`enableNewSalesforceMobileAppForTablet`	boolean	Indicates whether the org is enabled for the new Salesforce mobile app tablet experience (`true`) or not (`false`). If your org opted into the the new Salesforce mobile app during the pilot or in Winter '20, the default value is `true`. Otherwise, it's `false`. This field is available only for orgs that opted in to the new Salesforce mobile app during the pilot or in Winter '20. Available in API version 48.0 and later.
`enableOfflineDraftsEnabled`	boolean	Indicates whether users can create, edit, and delete records while offline in the Salesforce mobile app (`true`) or not (`false`). The default value is `true`. This option isn't available if `enableS1OfflinePref` is set to `false`. Available in API version 47.0 and later.
`enablePopulateNameManuallyInToday`	boolean	Indicates whether the user's name is shown on the Today page in the Salesforce mobile app (`true`) or not (`false`). The default value is `false`. Available in API version 47.0 and later.
`enableS1EncryptedStoragePref2`	boolean	Indicates whether the Salesforce mobile web uses secure and persistent browser caching to improve performance (`true`) or not (`false`). The default value is `true`. Available in API version 47.0 and later.
`enableS1OfflinePref`	boolean	Indicates whether users can access records offline in the Salesforce mobile app (`true`) or not (`false`). This option is set to `true` the first time someone in your org installs one of the Salesforce downloadable apps.

Field	Field Type	Description
		Available in API version 47.0 and later. However, offline access isn't supported in all versions of the downloadable mobile apps. Users must have version 10.0 or later of the Salesforce for Android app or the Salesforce for iOS app. Offline access isn't available for the Salesforce mobile web.
`touchMobile` (Deprecated)	TouchMobileSettings	Deprecated in API version 46.0. The settings for devices running Salesforce Touch.

ChatterMobileSettings

These fields are deprecated as of API version 46.0. Represents your organization's Chatter Mobile settings.

Field	Field Type	Description
`IPadAuthorized`	boolean	Indicates whether iPad devices are enabled for Chatter Mobile (`true`) or not (`false`).
`IPhoneAuthorized`	boolean	Indicates whether iPhone devices are enabled for Chatter Mobile (`true`) or not (`false`).
`androidAuthorized`	boolean	Indicates whether Android devices are enabled for Chatter Mobile (`true`) or not (`false`).
`blackBerryAuthorized`	boolean	Indicates whether Blackberry devices are enabled for Chatter Mobile (`true`) or not (`false`).
`enableChatterMobile`	boolean	Indicates whether Chatter Mobile has been enabled for your organization (`true`) or not (`false`). Note: Setting this field to `true` enables you to set all the other ChatterMobile settings. If you change this setting from `true` to `false`, and also try to change any of the other ChatterMobile settings, your deployment will fail with an error.
`enablePushNotifications`	boolean	Indicates whether Chatter push notifications have been enabled for your organization (`true`) or not (`false`)

Field	Field Type	Description
`sessionTimeout`	MobileSessionTimeout (enumeration of type string)	The length of time after which users without activity are prompted to log out or continue working. Valid values are: • `Never` • `OneMinute` • `FiveMinutes` • `TenMinutes` • `ThirtyMinutes`

DashboardMobileSettings

These fields are deprecated. Represents your organization's Mobile Dashboards iPad app settings.

Field	Field Type	Description
`enableDashboardIPadApp`	boolean	Indicates whether Mobile Dashboards iPad app has been enabled for your organization (`true`) or not (`false`).

TouchMobileSettings

These fields are deprecated as of API version 46.0. Salesforce Touch has been upgraded to the Salesforce mobile app.

Field	Field Type	Description
`enableTouchBrowserIPad`	boolean	Indicates whether your organization has the Salesforce Touch mobile browser app enabled (`true`) or not (`false`).
`enableTouchAppIPad`	boolean	Indicates whether your organization has the Salesforce Touch downloadable app enabled (`true`) or not (`false`)

Declarative Metadata Sample Definition

This is a sample `mobile.settings` metadata file.

```
<?xml version="1.0" encoding="UTF-8"?>
<MobileSettings xmlns="http://soap.sforce.com/2006/04/metadata">
    <dashboardMobile>
        <enableDashboardIPadApp>true</enableDashboardIPadApp>
    </dashboardMobile>
    <enableLightningOnMobile>false</enableLightningOnMobile>
    <enableImportContactFromDevice>true</enableImportContactFromDevice>
    <enableOfflineDraftsEnabled>true</enableOfflineDraftsEnabled>
    <enableS1EncryptedStoragePref2>true</enableS1EncryptedStoragePref2>
    <enableS1OfflinePref>true</enableS1OfflinePref>
</MobileSettings>
```

Wildcard Support in the Manifest File

The wildcard character `*` (asterisk) in the `package.xml` manifest file doesn't apply to metadata types for feature settings. The wildcard applies only when retrieving all settings, not for an individual setting. For details, see Settings. For information about using the manifest file, see Deploying and Retrieving Metadata with the Zip File.

MyDomainSettings

Represents your org's My Domain settings. My Domain is a Salesforce Identity feature that lets you create your own subdomain to better manage login and authentication. With a My Domain subdomain, you replace the instance name that Salesforce assigned you, such as `https://na00/salesforce.com` with a name that you choose, such as `https://`*`yourcompany`*`.my.salesforce.com`. With My Domain, you can create a custom login page and custom login policies to control how users are authenticated. You can also set up single sign-on (SSO) and social sign-on, work in multiple orgs at the same time, and more. This type extends the Metadata metadata type and inherits its `fullName` field.

File Suffix and Directory Location

MyDomainSettings values are stored in a single file named `MyDomain.settings` in the `settings` directory. The `.settings` files are different from other named components because there is only one settings file for each settings component.

Version

MyDomainSettings components are available in API version 47.0 and later.

Fields

Field Name	Field Type	Description
`canOnlyLoginWithMyDomainUrl`	boolean	If `true`, users must use their My Domain URL to log in. If `false` (default), users can also log in using their instance Salesforce URL, `https://`*`instance`*`.login.salesforce.com`, and through the login pool.
`doesApiLoginRequireOrgDomain`	boolean	If `true`, users must use their My Domain URL to access the Salesforce API. If `false` (default), users can also access the Salesforce API using their generic Salesforce page, `https://`*`instance`*`.login.salesforce.com`.
`enableNativeBrowserForAuthOnAndroid`	boolean	If `true`, use the native browser for authentication of Android mobile apps. Default is `false`.
`enableNativeBrowserForAuthOnIos`	boolean	If `true`, use the native browser for authentication of iOS mobile apps. Default is `false`.

Field Name	Field Type	Description
useStabilizedMyDomainHostnames	boolean	Indicates whether the instance name is hidden in My Domain URLs for Visualforce, Experience Builder, Site.com Studio, and Content Files (`true`) or not (`false`). This field has a default value of `false`. For example, `mydomain--c.na44.visual.force.com` becomes `mydomain--c.visualforce.com` when this field is set to `true`.
useStabilizedSandboxMyDomainHostnames	boolean	Indicates whether the instance name is hidden in My Domain URLs for sandboxes orgs (`true`) or not (`false`). This field has a default value of `false`. For example, `mydomain--test.cs5.my.salesforce.com` becomes `mydomain--test.my.salesforce.com` when this field is set to true.

Declarative Metadata Sample Definition

The following is an example of a MyDomainSettings component.

```
<?xml version="1.0" encoding="UTF-8"?>
<MyDomainSettings xmlns="http://soap.sforce.com/2006/04/metadata">
    <canOnlyLoginWithMyDomainUrl>false</canOnlyLoginWithMyDomainUrl>
    <doesApiLoginRequireOrgDomain>false</doesApiLoginRequireOrgDomain>
    <enableNativeBrowserForAuthOnAndroid>false</enableNativeBrowserForAuthOnAndroid>
    <enableNativeBrowserForAuthOnIos>false</enableNativeBrowserForAuthOnIos>
    <useStabilizedMyDomainHostnames>true</useStabilizedMyDomainHostnames>
    <useStabilizedSandboxMyDomainHostnames>false</useStabilizedSandboxMyDomainHostnames>
</MyDomainSettings>
```

The following is an example `package.xml` that references the previous definition.

```
<?xml version="1.0" encoding="UTF-8"?>
<Package xmlns="http://soap.sforce.com/2006/04/metadata">
    <types>
        <members>MyDomain</members>
        <name>Settings</name>
    </types>
    <version>47.0</version>
</Package>
```

NameSettings

Enables or disables the formal name, middle name, and suffix attributes for the following person objects: Contact, Lead, Person Account, and User.This type extends the Metadata metadata type and inherits its `fullName` field.

File Suffix and Directory Location

NameSettings values are stored in a single file named `Name.settings` in the `settings` folder. The `.settings` files are different from other named components because there is only one settings file for each settings component.

Version

NameSettings components are available in API version 31.0 and later.

Fields

Field Name	Field Type	Description
`enableFormalName`	boolean	Indicates whether informal names are enabled (`true`) or disabled (`false`) for person objects. Available in API version 48.0 and later.
`enableMiddleName`	boolean	Indicates whether middle names are enabled (`true`) or disabled (`false`) for person objects.
`enableNameSuffix`	boolean	Indicates whether suffixes are enabled (`true`) or disabled (`false`) for person objects.

Declarative Metadata Sample Definition

The following is an example of a NameSettings component.

```
<?xml version="1.0" encoding="UTF-8"?>
<NameSettings xmlns="http://soap.sforce.com/2006/04/metadata">
    <enableInformalName>false</enableInformalName>
    <enableMiddleName>true</enableMiddleName>
    <enableNameSuffix>false</enableNameSuffix>
</NameSettings>
```

The following is an example `package.xml` manifest that references the NameSettings definitions.

```
<?xml version="1.0" encoding="UTF-8"?>
<Package xmlns="http://soap.sforce.com/2006/04/metadata">
    <types>
        <members>Name</members>
        <name>Settings</name>
    </types>
    <version>31.0</version>
</Package>
```

Wildcard Support in the Manifest File

The wildcard character `*` (asterisk) in the `package.xml` manifest file doesn't apply to metadata types for feature settings. The wildcard applies only when retrieving all settings, not for an individual setting. For details, see Settings. For information about using the manifest file, see Deploying and Retrieving Metadata with the Zip File.

NotificationsSettings

Represents an organization's mobile settings.

In the package manifest, all organization settings metadata types are accessed using the "Settings" name. See Settings for more details.

Declarative Metadata File Suffix and Directory Location

NotificationsSettings values are stored in a single file named `Notifications.settings` in the `settings` directory. The `.settings` files are different from other named components because there is only one settings file for each settings component.

Version

Mobile settings are available in API version 46.0 and later.

Fields

Field	Field Type	Description
`enableMobileAppPushNotifications`	boolean	Indicates whether mobile push notifications are enabled.
`enableNotifications`	boolean	Indicates whether notifications are enabled.

Declarative Metadata Sample Definition

This is a sample `notifications.settings` metadata file.

```
<NotificationsSettings xmlns="http://soap.sforce.com/2006/04/metadata">
    <enableMobileAppPushNotifications>true</enableMobileAppPushNotifications>
    <enableNotifications>true</enableNotifications>
</NotificationsSettings>
```

Wildcard Support in the Manifest File

The wildcard character `*` (asterisk) in the `package.xml` manifest file doesn’t apply to metadata types for feature settings. The wildcard applies only when retrieving all settings, not for an individual setting. For details, see Settings. For information about using the manifest file, see Deploying and Retrieving Metadata with the Zip File.

ObjectLinkingSettings (Beta)

Represents the channel-object linking settings for an org. This type extends the Metadata metadata type and inherits its `fullName` field.

Note: As a beta feature, Channel-Object Linking is a preview and isn’t part of the “Services” under your master subscription agreement with Salesforce. Use this feature at your sole discretion, and make your purchase decisions only on the basis of generally available products and features. Salesforce doesn’t guarantee general availability of this feature within any particular time frame or at all, and we can discontinue it at any time. This feature is for evaluation purposes only, not for production use. It’s offered as is and isn’t supported, and Salesforce has no liability for any harm or damage arising out of or in connection with it. All restrictions, Salesforce reservation of rights, obligations concerning the Services, and terms for related Non-Salesforce Applications and Content apply equally to your use of this feature. For information on enabling this feature, contact Salesforce.

In the package manifest, all organization settings metadata types are accessed using the “Settings” name. See Settings for more details.

File Suffix and Directory Location

ObjectLinkingSettings values are stored in the `ObjectLinking.settings` file in the `settings` folder. The `.settings` files are different from other named components because there is only one settings file for each settings component.

Version

ObjectLinkingSettings components are available in API version 47.0 and later.

Fields

Field Name	Field Type	Description
`enableObjectLinking`	boolean	Indicates whether Channel-Object Linking is enabled, allowing you to link channel interactions to objects such as Contacts. The default value is `false`.

Declarative Metadata Sample Definition

The following is an example of an ObjectLinkingSettings component.

```
<?xml version="1.0" encoding="UTF-8"?>
<ObjectLinkingSettings xmlns="http://soap.sforce.com/2006/04/metadata">
    <enableObjectLinking>true</enableObjectLinking>
</ObjectLinkingSettings>
```

The following is an example `package.xml` that references the previous definition.

```
<?xml version="1.0" encoding="UTF-8"?>
<Package xmlns="http://soap.sforce.com/2006/04/metadata">
    <types>
        <members>ObjectLinking</members>
        <name>Settings</name>
    </types>
    <version>47.0</version>
</Package>
```

OmniChannelSettings

Represents the Omni-Channel settings for an org. This type extends the Metadata metadata type and inherits its `fullName` field.

In the package manifest, all organization settings metadata types are accessed using the “Settings” name. See Settings for more details.

File Suffix and Directory Location

OmniChannelSettings values are stored in the `OmniChannel.settings` file in the `settings` folder. The `.settings` files are different from other named components because there is only one settings file for each settings component.

Version

OmniChannelSettings components are available in API version 44.0 and later.

Fields

Field Name	Field Type	Description
`enableOmniAutoLoginPrompt`	boolean	Indicates whether to display a login confirmation upon loading a console with Omni-Channel. The default value is `false`. When `true`, the console displays a prompt before logging into Omni-Channel when an agent opens another Omni-Channel console in a different tab or window or refreshes the current tab. The agent is logged out of Omni-Channel on other consoles and any ongoing conversations are ended. Available in API version 47.0 and later.
`enableOmniChannel`	boolean	Indicates whether Omni-Channel is enabled, giving you access to the objects required to set up the feature in your org. The default value is `false`.
`enableOmniSecondaryRoutingPriority`	boolean	Indicates whether Secondary Routing Priority is enabled in your org. The default value is `false`. Available in API version 47.0 and later.
`enableOmniSkillsRouting`	boolean	Indicates whether skills-based routing is enabled in your org. The default value is `false`.

Declarative Metadata Sample Definition

The following is an example of a OmniChannelSettings component.

```
<?xml version="1.0" encoding="UTF-8"?>
<OmniChannelSettings xmlns="http://soap.sforce.com/2006/04/metadata">
    <enableOmniChannel>true</enableOmniChannel>
    <enableOmniAutoLoginPrompt>true</enableOmniAutoLoginPrompt>
    <enableOmniSecondaryRoutingPriority>true</enableOmniSecondaryRoutingPriority>
    <enableOmniSkillsRouting>true</enableOmniSkillsRouting>
</OmniChannelSettings>
```

The following is an example `package.xml` that references the previous definition.

```
<?xml version="1.0" encoding="UTF-8"?>
<Package xmlns="http://soap.sforce.com/2006/04/metadata">
    <types>
        <members>OmniChannel</members>
        <name>Settings</name>
    </types>
    <version>44.0</version>
</Package>
```

Wildcard Support in the Manifest File

The wildcard character `*` (asterisk) in the `package.xml` manifest file doesn't apply to metadata types for feature settings. The wildcard applies only when retrieving all settings, not for an individual setting. For details, see Settings. For information about using the manifest file, see Deploying and Retrieving Metadata with the Zip File.

OpportunityInsightsSettings

Represents an org's Einstein Opportunity Insights settings. This setting controls features that give you relevant updates about your opportunities. This type extends the Metadata metadata type and inherits its `fullName` field.

In the package manifest, all organization settings metadata types are accessed using the "Settings" name. See Settings for more details.

File Suffix and Directory Location

OpportunityInsightsSettings values are stored in the `OpportunityInsights.settings` file in the `settings` folder. The `.settings` files are different from other named components because there is only one settings file for each settings component.

Version

OpportunityInsightsSettings is available in API versions 48.0 and later.

Fields

Field Name	Field Type	Description
`enableOpportunityInsights`	boolean	Indicates whether Einstein Opportunity Insights is enabled (`true`) or not (`false`). The default value is `false`.

Declarative Metadata Sample Definition

The following is an example of the OpportunityInsights.settings file:

```
<?xml version="1.0" encoding="UTF-8"?>
<OpportunityInsightsSettings xmlns="http://soap.sforce.com/2006/04/metadata">
 <enableOpportunityInsights>true</enableOpportunityInsights>
</OpportunityInsightsSettings>
```

Example Package Manifest

The following is an example package manifest used to deploy or retrieve the OpportunityInsights settings metadata:

```
<?xml version="1.0" encoding="UTF-8"?>
<Package xmlns="http://soap.sforce.com/2006/04/metadata">
 <types>
  <members>OpportunityInsights</members>
  <name>Settings</name>
 </types>
 <version>29.0</version>
</Package>
```

Wildcard Support in the Manifest File

The wildcard character * (asterisk) in the `package.xml` manifest file doesn't apply to metadata types for feature settings. The wildcard applies only when retrieving all settings, not for an individual setting. For details, see Settings. For information about using the manifest file, see Deploying and Retrieving Metadata with the Zip File.

OpportunitySettings

Represents org preferences for features such as automatic opportunity updates and similar-opportunity filters. This type extends the Metadata metadata type and inherits its `fullName` field.

Use opportunity settings to control the actions that users can perform on their opportunities.

In the package manifest, all organization settings metadata types are accessed using the "Settings" name. See Settings for more details.

File Suffix and Directory Location

Opportunities values are stored in a single file named `Opportunity.settings` in the `settings` directory of the corresponding package directory. The `.settings` files are different from other named components because there is only one settings file for each settings component.

Version

OpportunitySettings is available in API version 28.0 and later.

Fields

Field Name	Field Type	Description
`autoActivateNewReminders`	boolean	Automatically uses scheduled updates for new opportunities.
`customizableProductSchedulesEnabled`	boolean	Lets Salesforce admins customize product schedules by using custom fields, validation rules, and Apex triggers on the LineItemSchedule object. This field is available in API version 46.0 and later. Note: If customizable product schedules are enabled, you can use custom fields in default schedules and customize their layout, but Apex triggers or validation rules that you apply to default schedules are bypassed.
`doesAutoAddSplitOwnerAsOpportunityTeamMember`	boolean	If `true`, members are automatically added to opportunity teams when a user edits the opportunity splits. If `false`, users must use the opportunity team's related list to add members. Default value is `true` for Enterprise, Performance, Developer, and Ultimate Editions. Available in API version 47.0 and later.
`doesEnforceStandardOpportunitySaveLogic`	boolean	Enforces standard validation and triggers for opportunity products and opportunity product schedules. Default value is `true`. Can't be set to `false`. Available in API version 47.0 and later.

Field Name	Field Type	Description
`enableUpdateReminders`	boolean	Lets users enable automatic, scheduled updates on opportunities.
`enableFindSimilarOpportunities`	boolean	Lets users see related or similar existing opportunities.
`enableOpportunityFieldHistoryTracking`	boolean	Enables history tracking for the opportunity field. For more information, see “Field History Tracking” in Salesforce Help. Default value is `true`. Available in API version 47.0 and later.
`enableOpportunityInsightsInMobile`	boolean	Indicates whether a user can see Einstein Opportunity Insights on their mobile device `(true)` or not `(false)`. Einstein Opportunity Insights includes predictions about which deals are likely to be won, reminders to follow up, and notifications when key moments in a deal take place. Available in API version 47.0 and later.
`enableOpportunityTeam`	boolean	Lets users associate team members with opportunities.
`findSimilarOppFilter`	FindSimilarOppFilter on page 973	Defines parameters for similar opportunities.
`promptToAddProducts`	boolean	Prompts users to add related products to an opportunity.

FindSimilarOppFilter

Defines whether to match by entire columns or fields.

Field	Field Type	Description
`similarOpportunitiesDisplayColumns`	string	The columns to compare.
`similarOpportunitiesMatchFields`	string	The fields to compare.

Declarative Metadata Sample Definition

The following is an example of the package file.

```
<?xml version="1.0" encoding="UTF-8"?>
<Package xmlns="http://soap.sforce.com/2006/04/metadata">
    <types>
        <members>Opportunity</members>
        <name>Settings</name>
    </types>
    <version>28.0</version>
</Package>
```

The package file references the following Opportunity.settings file.

```
<?xml version="1.0" encoding="UTF-8"?>
<OpportunitySettings xmlns="http://soap.sforce.com/2006/04/metadata">
  <autoActivateNewReminders>true</autoActivateNewReminders>
  <customizableProductSchedulesEnabled>false</customizableProductSchedulesEnabled>
```

```
<doesAutoAddSplitOwnerAsOpportunityTeamMember>true</doesAutoAddSplitOwnerAsOpportunityTeamMember>

  <doesEnforceStandardOpportunitySaveLogic>true</doesEnforceStandardOpportunitySaveLogic>

  <enableFindSimilarOpportunities>true</enableFindSimilarOpportunities>
  <findSimilarOppFilter>
    <similarOpportunitiesMatchFields>OPPORTUNITY.Account</similarOpportunitiesMatchFields>


<similarOpportunitiesMatchFields>OPPORTUNITY.OpportunityCompetitors</similarOpportunitiesMatchFields>

    <similarOpportunitiesMatchFields>CustomField__c</similarOpportunitiesMatchFields>
    <similarOpportunitiesDisplayColumns>CustomField__c</similarOpportunitiesDisplayColumns>

  </findSimilarOppFilter>
  <enableOpportunityFieldHistoryTracking>true</enableOpportunityFieldHistoryTracking>
  <enableOpportunityInsightsInMobile>true</enableOpportunityInsightsInMobile>
  <enableOpportunityTeam>true</enableOpportunityTeam>
  <enableUpdateReminders>true</enableUpdateReminders>
  <promptToAddProducts>true</promptToAddProducts>
</OpportunitySettings>
```

Wildcard Support in the Manifest File

The wildcard character `*` (asterisk) in the `package.xml` manifest file doesn't apply to metadata types for feature settings. The wildcard applies only when retrieving all settings, not for an individual setting. For details, see Settings. For information about using the manifest file, see Deploying and Retrieving Metadata with the Zip File.

OrderManagementSettings

Represents options for the Salesforce Order Management product.This type extends the Metadata metadata type and inherits its `fullName` field.

In the package manifest, all organization settings metadata types are accessed using the "Settings" name. See Settings for more details.

File Suffix and Directory Location

OrderManagementSettings values are stored in the `OrderManagement.settings` file in the `settings` directory. The `.settings` files are different from other named components because there is only one settings file for each settings component.

Version

Order Management settings are available in API version 48 and later.

Special Access Rules

This metadata is only used by developers and customers using Salesforce Order Management.

Fields

Field Name	Field Type	Description
`enableB2CIntegration`	boolean	Indicates whether Order Management is allowed to accept order data from B2C Commerce (`true`) or not (`false`).
`enableB2CSelfService`	boolean	Indicates whether Order Management is allowed to provide order data to B2C Commerce (`true`) or not (`false`).
`enableOrderManagement`	boolean	Indicates whether Order Management features are enabled (`true`) or not (`false`).

Declarative Metadata Sample Definition

The following is an example of an OrderManagementSettings component.

```
<?xml version="1.0" encoding="UTF-8"?>
<OrderManagementSettings xmlns="http://soap.sforce.com/2006/04/metadata"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
    <enableOrderManagement>true</enableOrderManagement>
    <enableB2CIntegration>true</enableB2CIntegration>
    <enableB2CSelfService>true</enableB2CSelfService>
</OrderManagementSettings>
```

The following is an example `package.xml` that references the previous definition.

```
<?xml version="1.0" encoding="UTF-8"?>
<Package xmlns="http://soap.sforce.com/2006/04/metadata">
    <types>
        <members>OrderManagement</members>
        <name>Settings</name>
    </types>
    <version>48.0</version>
</Package>
```

Wildcard Support in the Manifest File

The wildcard character `*` (asterisk) in the `package.xml` manifest file doesn’t apply to metadata types for feature settings. The wildcard applies only when retrieving all settings, not for an individual setting. For details, see Settings. For information about using the manifest file, see Deploying and Retrieving Metadata with the Zip File.

OrderSettings

Represents order settings. This type extends the Metadata metadata type and inherits its `fullName` field.

For more information, see “Set Up Orders” in Salesforce Help.

In the package manifest, all organization settings metadata types are accessed using the “Settings” name. See Settings for more details.

File Suffix and Directory Location

There is one OrderSettings component in a file named `Order.settings` in the `settings` folder. The `.settings` files are different from other named components because there is only one settings file for each settings component.

Version

OrderSettings components are available in API version 30.0 and later.

Fields

Field Name	Field Type	Description
`enableEnhancedCommerceOrders`	boolean	Indicates whether enhanced commerce orders are enabled for the org (`true`) or not (`false`). This preference is available only in orgs with the Salesforce Order Management license. Default value is `false.` Available in API versions 48.0 and later.
`enableNegativeQuantity`	boolean	Indicates whether users in the org can add order products with quantities of less than zero (`true`) or not (`false`). To enable this preference, `enableOrders` must be set to `true`.
`enableOrders`	boolean	Indicates whether orders are enabled for the org (`true`) or not (`false`).
`enableReductionOrders`	boolean	Indicates whether reduction orders are enabled for the org (`true`) or not (`false`). For more information, see “Reduction Orders” in Salesforce Help. To enable this preference, `enableOrders` must be set to `true`.
`enableZeroQuantity`	boolean	Indicates whether users in the org can add order products with quantities of zero (`true`) or not (`false`). Default value is `false`. To enable this preference, `enableOrders` must be set to `true`. Available in API version 42.0 and later.

Declarative Metadata Sample Definition

This is a sample OrderSettings component.

```
<?xml version="1.0" encoding="UTF-8"?>
<OrderSettings xmlns="http://soap.sforce.com/2006/04/metadata">
    <enableEnhancedCommerceOrders>true</enableEnhancedCommerceOrders/>

    <enableReductionOrders>false</enableReductionOrders>
    <enableNegativeQuantity>true</enableNegativeQuantity>
</OrderSettings>
```

The following is an example `package.xml` that references the previous definition.

```
<?xml version="1.0" encoding="UTF-8"?>
<Package xmlns="http://soap.sforce.com/2006/04/metadata">
    <types>
        <members>Order</members>
        <name>Settings</name>
    </types>
    <version>30.0</version>
</Package>
```

Wildcard Support in the Manifest File

The wildcard character `*` (asterisk) in the `package.xml` manifest file doesn't apply to metadata types for feature settings. The wildcard applies only when retrieving all settings, not for an individual setting. For details, see Settings. For information about using the manifest file, see Deploying and Retrieving Metadata with the Zip File.

OrgPreferenceSettings

Removed in API version 48.0. Represents the unique org preference settings in a Salesforce org.

This type extends the Metadata metadata type and inherits its `fullName` field.

File Suffix and Directory Location

OrgPreferenceSettings values are stored in the `OrgPreference.settings` file in the settings directory. The `.settings` files are different from other named components because there is only one settings file for each settings component.

Version

OrgPreferenceSettings components are available in API versions 37.0 to 47.0.

OrgPreferenceSettings is deprecated in API version 47.0 and removed in API version 48.0. In API version 47.0, most of the settings supported in the `preferences` field were made available in the form of Boolean fields on other Settings types. For example, in API version 47.0 and later, you can enable and disable the `CompileOnDeploy` preference by using the `enableCompileOnDeploy` field on the ApexSettings type.

Fields

Field Name	Field Type	Description
`preferences`	OrganizationSettingsDetail[]	The preferences associated with the org settings. In the following list of preferences, click hyperlinked preference names to go to the topic for the Settings type that contains that preference. If there is no link, the preference hasn't been moved to another Settings type. • `AnalyticsSharingEnable` (available in API version 40.0 and later) • `ApexApprovalLockUnlock` • `AsyncSaveEnabled` (available in API versions 40.0 to 46.0)

Field Name	Field Type	Description
		• ChatterEnabled • CompileOnDeploy (available in API version 43.0 and later) • ConsentManagementEnabled (available in API version 45.0 and later) • EnhancedEmailEnabled • EventLogWaveIntegEnabled • LoginForensicsEnabled • NetworksEnabled (available in API version 40.0 and later) • NotesReservedPref01 • OfflineDraftsEnabled • PathAssistantsEnabled • S1DesktopEnabled Note: After it is enabled, S1DesktopEnabled can't be disabled in any version of the API. • S1EncryptedStoragePref2 • S1OfflinePref • ScratchOrgManagementPref on page 895 (available in API version 41.0 and later) • SendThroughGmailPref • SocialProfilesEnable • Translation (available in API version 40.0 and later) • VoiceEnabled Note: The VoiceEnabled preference isn't being moved to another metadata type. If you want to use it in a scratch org in API version 48.0 and later, you can enable it as a scratch org feature.

OrganizationSettingsDetail

Field Name	Field Type	Description
settingName	string	The name of the setting. For example, "S1EncryptedStoragePref2."
settingValue	boolean	Indicates whether the setting is enabled (true) or not (false).

Declarative Metadata Sample Definition

The following is an example of a OrgPreferenceSettings component. The example shows only the `preferences` values that are supported but not yet available as fields on another Settings type in API version 47.0.

```
<?xml version="1.0" encoding="UTF-8"?>
<OrgPreferenceSettings xmlns="http://soap.sforce.com/2006/04/metadata">
    <preferences>
        <settingName>AnalyticsSharingEnable</settingName>
        <settingValue>true</settingValue>
    </preferences>
    <preferences>
        <settingName>NetworksEnabled</settingName>
        <settingValue>true</settingValue>
    </preferences>
    <preferences>
        <settingName>NotesReservedPref01</settingName>
        <settingValue>false</settingValue>
    </preferences>
    <preferences>
        <settingName>ScratchOrgManagementPref</settingName>
        <settingValue>true</settingValue>
    </preferences>
    <preferences>
        <settingName>VoiceEnabled</settingName>
        <settingValue>false</settingValue>
    </preferences>
</OrgPreferenceSettings>
```

Wildcard Support in the Manifest File

The wildcard character `*` (asterisk) in the `package.xml` manifest file doesn't apply to metadata types for feature settings. The wildcard applies only when retrieving all settings, not for an individual setting. For details, see Settings. For information about using the manifest file, see Deploying and Retrieving Metadata with the Zip File.

OrgSettings

Represents the settings for org-wide functionality that isn't associated with any specific feature.This type extends the Metadata metadata type and inherits its `fullName` field.

In the package manifest, all organization settings metadata types are accessed using the "Settings" name. See Settings for more details.

File Suffix and Directory Location

A OrgSettings component file has the suffix `.settings` and is stored in the `settings` directory. The `.settings` files are different from other named components because there is only one settings file for each settings component.

Version

OrgSettings components are available in API version 46.0 and later.

Fields

Field Name	Field Type	Descriptions
`enableCustomerSuccessPortal`	boolean	Indicates whether Customer Portal is enabled (`true`) or not (`false`).
`enableExtendedMailMerge`	boolean	Indicates whether the Salesforce Classic product, Extended Mail Merge, is enabled (`true`) or not (`false`). Use Extended Mail Merge to generate large volumes of Microsoft Word documents — such as form letters or address labels — from Word document templates. Default value is `false`. Available in API version 47.0 only.
`enableManageSelfServiceUsers`	boolean	Indicates whether mass management of self-service users is enabled through the Self-Service Portal (`true`) or not (`false`).
`saveMailMergeDocsAsSalesforceDocs`	boolean	Indicates whether users can save all documents that are generated using Extended Mail Merge as Salesforce documents (`true`) or not (`false`). If (`false`), Extended Mail Merge documents that are smaller than 3 MB are emailed to users who run the merge. Default value is (`false`). Available in API version 47.0 only.

Declarative Metadata Sample Definition

The following is an example of a OrgSettings component.

```
<?xml version="1.0" encoding="UTF-8"?>
<OrgSettings xmlns="http://soap.sforce.com/2006/04/metadata"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
    <enableCustomerSuccessPortal>false</enableCustomerSuccessPortal>
    <enableIncludeContractStatus>false</enableIncludeContractStatus>
    <enableMakeDeploymentsMandatory>true</enableMakeDeploymentsMandatory>
    <enableManageSelfServiceUsers>false</enableManageSelfServiceUsers>
    <enableOrgFeedSentimentAnalysis>false</enableOrgFeedSentimentAnalysis>
    <enableRADeploymentAttributeOnly>true</enableRADeploymentAttributeOnly>
    <enableResetDivisionOnLogin xsi:nil="true"/>
</OrgSettings>
```

Example Package Manifest

The following is an example package manifest used to deploy or retrieve the org settings metadata for an organization:

```
<?xml version="1.0" encoding="UTF-8"?>
<Package xmlns="http://soap.sforce.com/2006/04/metadata">
    <types>
        <members>Org</members>
        <name>Settings</name>
    </types>
    <version>47.0</version>
</Package>
```

Wildcard Support in the Manifest File

The wildcard character `*` (asterisk) in the `package.xml` manifest file doesn't apply to metadata types for feature settings. The wildcard applies only when retrieving all settings, not for an individual setting. For details, see Settings. For information about using the manifest file, see Deploying and Retrieving Metadata with the Zip File.

PartyDataModelSettings

Represents an organization's party data model settings, including options around the Individual object and consent enablement. This type extends the Metadata metadata type and inherits its `fullName` field.

In the package manifest, all organization settings metadata types are accessed using the "Settings" name. See Settings for more details.

File Suffix and Directory Location

PartyDataModelSettings values are stored in the `PartyDataModel.settings` file in the `settings` directory. The `.settings` files are different from other named components because there is only one settings file for each settings component.

Version

PartyDataModelSettings is available in API version 47.0 and later.

Fields

Field Name	Field Type	Description
`enableAutoSelectIndividualOnMerge`	boolean	Indicates whether the most recently modified data privacy record for the Individual is retained when merging lead, contact, and person accounts (`true`) or users must manually determine which data privacy record to retain during the merge process (`false`). This field has a default value of `false`.
`enableConsentManagementEnabled`	boolean	Indicates whether data protection details are available in records (`true`) or not (`false`). This has a default value of `true`. Note: Setting this field to `false` purges all data protection details, such as privacy preferences and stored consent forms.
`enableIndividualAutoCreate`	boolean	Deprecated in API version 48.0 and later.

Declarative Metadata Sample Definition

The following is an example of a PartyDataModelSettings component.

```
<?xml version="1.0" encoding="UTF-8"?>
<PartyDataModelSettings xmlns="http://soap.sforce.com/2006/04/metadata">
    <enableAutoSelectIndividualOnMerge>true</enableAutoSelectIndividualOnMerge>
    <enableConsentManagementEnabled>true</enableConsentManagementEnabled>
</PartyDataModelSettings>
```

The following is an example `package.xml` that references the previous definition.

```
<?xml version="1.0" encoding="UTF-8"?>
<Package xmlns="http://soap.sforce.com/2006/04/metadata">
    <types>
        <members>PartyDataModel</members>
        <name>Settings</name>
    </types>
    <version>47.0</version>
</Package>
```

Wildcard Support in the Manifest File

The wildcard character `*` (asterisk) in the `package.xml` manifest file doesn't apply to metadata types for feature settings. The wildcard applies only when retrieving all settings, not for an individual setting. For details, see Settings. For information about using the manifest file, see Deploying and Retrieving Metadata with the Zip File.

PardotSettings

Represents Pardot settings in your Salesforce org. Pardot is a powerful B2B marketing automation solution that helps you create meaningful connections, generate more pipeline, and empower sales to close more deals. You can use these settings to configure how Pardot collects and displays data.This type extends the Metadata metadata type and inherits its `fullName` field.

In the package manifest, all organization settings metadata types are accessed using the "Settings" name. See Settings for more details.

File Suffix and Directory Location

PardotSettings is stored in one file named `Pardot.Settings` in the `settings` folder of the corresponding package directory. The `.settings` files are different from other named components because there is only one settings file for each settings component.

Version

PardotSettings is available in API version 47.0 and later.

Special Access Rules

This metadata type is available only to accounts with Pardot.

Fields

Field Name	Field Type	Description
`enableB2bmaAppEnabled`	boolean	This feature is deprecated and isn't used.
`enableEngagementHistoryDashboards`	boolean	Enable the Engagement History Dashboard and allow related Pardot data to be shared to campaign records in Salesforce (`true`) or not (`false`). The default value is `false`. If `enableEnagementHistoryDashboards` is disabled after being enabled, the Engagement History Dashboard is removed, but Pardot engagement data is retained and continues to update.

Field Name	Field Type	Description
enablePardotAppV1Enabled	boolean	Enable the Pardot Lightning App (true) or not (false). The default value is false.
enablePardotEnabled	boolean	Enable the Pardot Marketing Automation Pilot including Social Search (true) or not (false). The default value is false. This feature is deprecated and isn't used.
enableProspectActivityDataset	boolean	Enable the Prospect and Activity Dataset for B2B Marketing Automation apps (true) or not (false). When enableProspectActivityDataset is set to true, the datasets take some time to populate. Depending on how much data and what kind of licenses you have, enabling this preference can impact the account's row limit for Analytics. If enableProspectActivityDataset is disabled after being enabled: • The data that makes up the datasets is deleted. • The Prospect and Activity Dataset in existing B2B Marketing Automation apps stops getting updates. • The dataset is not available to add to new apps. • When apps are reconfigured, the dataset is deleted. Requires that enableEnagementHistoryDashboardsis set to true.

Declarative Metadata Sample Definition

The following is an example of a PardotSettings component.

```xml
<?xml version="1.0" encoding="UTF-8"?>
<PardotSettings xmlns="http://soap.sforce.com/2006/04/metadata">
    <enablePardotAppV1Enabled>true</enablePardotAppV1Enabled>
    <enableEngagementHistoryDashboards>true</enableEngagementHistoryDashboards>
    <enablePardotEnabled>true</enablePardotEnabled>
    <enableB2bmaAppEnabled>true</enableB2bmaAppEnabled>
    <enableProspectActivityDataset>true</enableProspectActivityDataset>
</PardotSettings>
```

The following is an example package.xml that references the previous definition.

```xml
<?xml version="1.0" encoding="UTF-8"?>
<Package xmlns="http://soap.sforce.com/2006/04/metadata">
    <types>
        <members>Pardot</members>
        <name>Settings</name>
    </types>
    <version>47</version>
</Package>
```

Wildcard Support in the Manifest File

The wildcard character `*` (asterisk) in the `package.xml` manifest file doesn't apply to metadata types for feature settings. The wildcard applies only when retrieving all settings, not for an individual setting. For details, see Settings. For information about using the manifest file, see Deploying and Retrieving Metadata with the Zip File.

PardotEinsteinSettings

Represents PardotEinsteinSettings. Use these settings to learn what factors drive your campaign performance, and get the best possible engagement score for your prospects. This type extends the Metadata metadata type and inherits its `fullName` field.

In the package manifest, all organization settings metadata types are accessed using the "Settings" name. See Settings for more details.

File Suffix and Directory Location

PardotEinsteinSettings values are stored in the `PardotEinstein.settings` file in the `settings` folder. The `.settings` files are different from other named components because there is only one settings file for each settings component.

Version

PardotEinsteinSettings is available in API versions 48.0 and later.

Fields

Field Name	Field Type	Description
`enableCampaignInsight`	boolean	Indicates whether Einstein Campaign Insights is enabled (`true`) or not (`false`). Einstein Campaign Insights helps you understand what factors drive campaign performance. The default value is `false`.
`enableEngagementScore`	boolean	Indicates whether Einstein Behavior Scoring is enabled (`true`) or not (`false`). Einstein Behavior Scoring identifies prospects whose behavior suggests that they are ready to buy, and scores them based on Einstein's engagement model. The default value is `false`.

Declarative Metadata Sample Definition

The following is an example of the PardotEinstein.settings file:

```
<?xml version="1.0" encoding="UTF-8"?>
<PardotEinsteinSettings xmlns="http://soap.sforce.com/2006/04/metadata">
 <enableCampaignInsight>true</enableCampaignInsight>
 <enableEngagementScore>true</enableEngagementScore>
</PardotEinsteinSettings>
```

Example Package Manifest

The following is an example package manifest used to deploy or retrieve the PardotEinstein settings metadata:

```
<?xml version="1.0" encoding="UTF-8"?>
<Package xmlns="http://soap.sforce.com/2006/04/metadata">
 <types>
  <members>PardotEinstein</members>
  <name>Settings</name>
 </types>
 <version>29.0</version>
</Package>
```

Wildcard Support in the Manifest File

The wildcard character * (asterisk) in the package.xml manifest file doesn't apply to metadata types for feature settings. The wildcard applies only when retrieving all settings, not for an individual setting. For details, see Settings. For information about using the manifest file, see Deploying and Retrieving Metadata with the Zip File.

PathAssistantSettings

Represents the Path preference setting. This type extends the Metadata metadata type and inherits its fullName field.

In the package manifest, all organization settings metadata types are accessed using the "Settings" name. See Settings for more details.

File Suffix and Directory Location

PathAssistantSettings components have the suffix .settings and are stored in the settings folder.

Version

PathAssistantSettings components are available in API version 34.0 and later.

Fields

Field Name	Field Type	Description
canOverrideAutoPathCollapseWithUserPref	boolean	Keeps a user's path expanded to show guidance and key fields on all their records. A user's path stays expanded until the user collapses it. To use this preference, Path must be enabled. Default value is false for all editions. When set to false, the user's path is collapsed when the page loads. Available in API version 47.0 and later.
pathAssistantEnabled	boolean	Determines whether the preference is enabled for Path. Default value is true for Enterprise Edition and false for other editions. Available in API version 35.0 and later.

Field Name	Field Type	Description
pathAssistantForOpportunityEnabled	boolean	Determines whether the preference is enabled for Path in Opportunity or not. Available in API version 34.0 and earlier.

Declarative Metadata Sample Definition

The following is an example of a PathAssistantSettings component.

```
<?xml version="1.0" encoding="UTF-8"?>
<PathAssistantSettings xmlns="http://soap.sforce.com/2006/04/metadata">
    <pathAssistantEnabled>true</pathAssistantEnabled>
   <canOverrideAutoPathCollapseWithUserPref>true</canOverrideAutoPathCollapseWithUserPref>
</PathAssistantSettings>
```

The following is an example `package.xml` that references the previous definition.

```
<?xml version="1.0" encoding="UTF-8"?>
<Package xmlns="http://soap.sforce.com/2006/04/metadata">
    <types>
        <members>PathAssistant</members>
        <name>Settings</name>
    </types>
    <version>API</version>
</Package
```

Wildcard Support in the Manifest File

The wildcard character `*` (asterisk) in the `package.xml` manifest file doesn't apply to metadata types for feature settings. The wildcard applies only when retrieving all settings, not for an individual setting. For details, see Settings. For information about using the manifest file, see Deploying and Retrieving Metadata with the Zip File.

PicklistSettings

Represents an org's picklist settings. These settings control the behavior of a picklist. This type extends the Metadata metadata type and inherits its `fullName` field.

In the package manifest, all organization settings metadata types are accessed using the "Settings" name. See Settings for more details.

File Suffix and Directory Location

PicklistSettings values are stored in a single file named `Picklist.settings` in the `settings` directory. The `.settings` files are different from other named components because there is only one settings file for each settings component.

Version

Picklist settings are available in API version 47.0 and later.

Fields

Field Name	Field Type	Description
isPicklistApiNameEditDisabled	boolean	While `true`, users, including admins with Customize Application permission, can't change the API name of a picklist field. Formulas reference a picklist's API name so that the formula continues to work even if the displayed name value changes. Prevent changes to the API name to protect the references to fields in formulas or during integrations, such as during a data import. The default is `false`.

Declarative Metadata Sample Definition

The following is a sample `picklist.settings` metadata file.

```
<?xml version="1.0" encoding="UTF-8"?>
<PicklistSettings xmlns="http://soap.sforce.com/2006/04/metadata">
  <isPicklistApiNameEditDisabled>true</isPicklistApiNameEditDisabled>
</PicklistSettings>
```

The following is an example `package.xml` manifest that references the previous definition.

```
<?xml version="1.0" encoding="UTF-8"?>
<Package xmlns="http://soap.sforce.com/2006/04/metadata">
 <types>
  <members>Picklist</members>
  <name>Settings</name>
 </types>
 <version>47.0</version>
</Package>
```

Wildcard Support in the Manifest File

The wildcard character `*` (asterisk) in the `package.xml` manifest file doesn't apply to metadata types for feature settings. The wildcard applies only when retrieving all settings, not for an individual setting. For details, see Settings. For information about using the manifest file, see Deploying and Retrieving Metadata with the Zip File.

PlatformEncryptionSettings

Represents an org's Platform Encryption settings, such as settings for available encryption schemes, permissions, encryption policy access, and which fields can be encrypted. This type extends the Metadata metadata type and inherits its `fullName` field.

In the package manifest, all organization settings metadata types are accessed using the "Settings" name. See Settings for more details.

File Suffix and Directory Location

PlatformEncryptionSettings values are stored in the `PlatformEncryption.settings` file in the `settings` folder. The `.settings` files are different from other named components because there is only one settings file for each settings component.

Version

PlatformEncryptionSettings is available in API versions 47.0 and later.

Special Access Rules

To enable and disable PlatformEncryptionSettings attributes, you need the Customize Application permission. Attributes that allow key management tasks require the Manage Encryption Keys permission. For a complete list of required permissions, read Which User Permissions Does Shield Platform Encryption Require?.

Fields

Field Name	Field Type	Description
`canEncryptManagedPackageFields`	boolean	Indicates whether users can enable encryption on custom fields in installed managed packages (`true`) or not (`false`).
`isUseHighAssuranceKeysRequired`	boolean	Indicates whether key management actions require a second form of authentication (`true`) or not (`false`). The second form of authentication can be an app like Salesforce Authenticator, a Yubikey, or other time-based one time password. The default value is `false`. Warning: When enabled, admins without two-factor authentication can't manage encryption key material.
`isMEKForEncryptionRequired`	boolean	Indicates whether encryption policy tasks, such as enabling encryption on fields, also require the Manage Encryption Keys permission (`true`) or not (`false`), in addition to those tasks' baseline permissions.
`enableDeterministEncryption`	boolean	Indicates whether customers apply the deterministic encryption scheme to supported fields (`true`) or not (`false`). The deterministic encryption scheme lets customers filter on encrypted data..
`enableEncryptFieldHistory`	boolean	Indicates whether the background encryption process applies the customer's active key material to field history and feed tracking values (`true`) or not (`false`). The default value is `false`. If `false`, background encryption processes apply active key material to all encrypted data except duplicates of that data stored in field history or feed tracking.
`enableEventBusEncryption`	boolean	Indicates whether events are encrypted at rest in the event bus (`true`) or not (`false`). The events include change data capture events and platform events. The default value is `false`. If `false`, events aren't encrypted and are stored in clear text in the event bus. Warning: Generate or upload key material of the Event Bus type before turning on the `enableEventBusEncryption` setting. If you don't, event payloads are blocked and aren't delivered.

Declarative Metadata Sample Definition

The following is an example of the PlatformEncryption.settings file:

```
<?xml version="1.0" encoding="UTF-8"  standalone="yes"?>
<PlatformEncryptionSettings xmlns="http://soap.sforce.com/2006/04/metadata">
    <canEncryptManagedPackageFields>true</canEncryptManagedPackageFields>
    <isUseHighAssuranceKeysRequired>true</isUseHighAssuranceKeysRequired>
    <isMEKForEncryptionRequired>true</isMEKForEncryptionRequired>
    <enableDeterministEncryption>true</enableDeterministEncryption>
    <enableEncryptFieldHistory>true</enableEncryptFieldHistory>
    <enableEventBusEncryption>true</enableEventBusEncryption>
</PlatformEncryptionSettings>
```

Example Package Manifest

The following is an example package manifest used to deploy or retrieve the Platform Encryption settings metadata for an organization:

```
<?xml version="1.0" encoding="UTF-8"?>
<Package xmlns="http://soap.sforce.com/2006/04/metadata">
    <types>
        <members>Platform Encryption</members>
        <name>Settings</name>
    </types>
    <version>47.0</version>
</Package>
```

Wildcard Support in the Manifest File

The wildcard character `*` (asterisk) in the `package.xml` manifest file doesn't apply to metadata types for feature settings. The wildcard applies only when retrieving all settings, not for an individual setting. For details, see Settings. For information about using the manifest file, see Deploying and Retrieving Metadata with the Zip File.

PredictionBuilderSettings

Represents the settings that determine how a user can interact with Einstein Prediction Builder. This type extends the Metadata metadata type and inherits its `fullName` field.

In the package manifest, all organization settings metadata types are accessed using the "Settings" name. See Settings for more details.

File Suffix and Directory Location

PredictionBuilderSettings values are stored in the PredictionBuilder.settings file in the settings directory. The .settings files are different from other named components in that each settings component has only one settings file.

Version

PredictionBuilderSettings components are available in API version 47.0 and later.

Special Access Rules

This type is available only if the Einstein Analytics Plus or Einstein Predictions license is enabled in your org.

Fields

Field Name	Field Type	Description
`enablePredictionBuilder`	boolean	Indicates whether Einstein Prediction Builder is enabled (`true`) or not (`false`).
`isPredictionBuilderStarted`	boolean	Indicates whether to display the predictions list view to the user (`true`) or not (`false`).

Declarative Metadata Sample Definition

This is a sample Prediction Builder settings file.

```
<?xml version="1.0" encoding="UTF-8"?>
<PredictionBuilderSettings xmlns="http://soap.sforce.com/2006/04/metadata">
    <isPredictionBuilderStarted>false</isPredictionBuilderStarted>
    <enablePredictionBuilder>false</enablePredictionBuilder>
</PredictionBuilderSettings>
```

Wildcard Support in the Manifest File

The wildcard character `*` (asterisk) in the `package.xml` manifest file doesn't apply to metadata types for feature settings. The wildcard applies only when retrieving all settings, not for an individual setting. For details, see Settings. For information about using the manifest file, see Deploying and Retrieving Metadata with the Zip File.

PrivacySettings

Represents an organization's settings for data privacy and consent management. This type extends the Metadata metadata type and inherits its `fullName` field.

In the package manifest, all organization settings metadata types are accessed using the "Settings" name. See Settings for more details.

File Suffix and Directory Location

PrivacySettings values are stored in the `Privacy.settings` file in the `settings` directory. The `.settings` files are different from other named components because there is only one settings file for each settings component.

Version

PrivacySettings components are available in API version 47.0 and later.

Special Access Rules

To use PrivacySettings, you need the Customize Application or Modify Data Classification user permission.

Fields

Field Name	Field Type	Description
enableConsentAuditTrail	boolean	Reserved for future use.
enableConsentEventStream	boolean	Reserved for future use.
enableDefaultMetadataValues	boolean	Indicates whether a default data sensitivity value is applied to all contacts, leads, person accounts, and users (`true`) or not (`false`). This field has a default value of `false`. Available in API version 47.0 and later.

Declarative Metadata Sample Definition

The following is an example of a PrivacySettings component.

```
<?xml version="1.0" encoding="UTF-8"?>
<PrivacySettings xmlns="http://soap.sforce.com/2006/04/metadata">
    <enableDefaultMetadataValues>false</enableDefaultMetadataValues>
</PrivacySettings>
```

The following is an example `package.xml` that references the previous definition.

```
<?xml version="1.0" encoding="UTF-8"?>
<Package xmlns="http://soap.sforce.com/2006/04/metadata">
    <types>
        <members>Privacy</members>
        <name>Settings</name>
    </types>
    <version>47.0</version>
</Package>
```

Wildcard Support in the Manifest File

The wildcard character `*` (asterisk) in the `package.xml` manifest file doesn't apply to metadata types for feature settings. The wildcard applies only when retrieving all settings, not for an individual setting. For details, see Settings. For information about using the manifest file, see Deploying and Retrieving Metadata with the Zip File.

ProductSettings

Represents organization preferences for quantity schedules, revenue schedules, and active flag interaction with prices. This type extends the Metadata metadata type and inherits its `fullName` field.

File Suffix and Directory Location

ProductSettings values are stored in a single file named `Product.settings` in the `settings` directory of the corresponding package directory. The `.settings` files are different from other named components because there is only one settings file for each settings component.

Version

ProductSettings is available in API version 28.0 and later.

Fields

Field Name	Field Type	Description
`enableCascadeActivateToRelatedPrices`	boolean	When changing active flag on a product record, automatically updates active flag on related prices.
`enableMySettings`	boolean	Moves users' personal settings pages from Setup to a separate My Settings pane (`true`) or not (`false`). When set to (`true`), Salesforce makes a reorganized Setup pane accessible to admins via one click in the header. This setting affects all users in your organization. The default is `true`. Available in API version 47.0 and later.
`enableQuantitySchedule`	boolean	Enables quantity schedules for products.
`enableRevenueSchedule`	boolean	Enables revenue schedules for products.

Declarative Metadata Sample Definition

The following is an example of the package file.

```
<?xml version="1.0" encoding="UTF-8"?>
<Package xmlns="http://soap.sforce.com/2006/04/metadata">
    <types>
        <members>Product</members>
        <name>Settings</name>
    </types>
    <version>28.0</version>
</Package>
```

The package file references the following Product.settings file.

```
<?xml version="1.0" encoding="UTF-8"?>
<ProductSettings xmlns="http://soap.sforce.com/2006/04/metadata">
    <enableCascadeActivateToRelatedPrices>true</enableCascadeActivateToRelatedPrices>
    <enableQuantitySchedule>false</enableQuantitySchedule>
    <enableRevenueSchedule>false</enableRevenueSchedule>
</ProductSettings>
```

Wildcard Support in the Manifest File

The wildcard character `*` (asterisk) in the `package.xml` manifest file doesn't apply to metadata types for feature settings. The wildcard applies only when retrieving all settings, not for an individual setting. For details, see Settings. For information about using the manifest file, see Deploying and Retrieving Metadata with the Zip File.

QuoteSettings

Represents an org's quotes settings, such as enabling quotes or creating quotes without an associated opportunity. This type extends the Metadata metadata type and inherits its `fullName` field.

In the package manifest, all organization settings metadata types are accessed using the "Settings" name. See Settings for more details.

File Suffix and Directory Location

QuoteSettings values are stored in a single file named `Quote.settings` in the `settings` directory of the corresponding package directory. The `.settings` files are different from other named components because there is only one settings file for each settings component.

Version

QuoteSettings is available in API version 28.0 and later.

Fields

Field Name	Field Type	Description
`enableQuote`	boolean	When set to `true`, users can access Quotes.
`enableQuotesWithoutOppEnabled`	boolean	When set to `true`, users can create quotes independently of an opportunity. For example, a user can create a quote for budgeting purposes, before creating the Opportunity. Default value is `false`. When set to `false`, users can only create quotes from an Opportunity. Before setting to `false`, delete any quotes that do not have opportunities. Available in API version 47.0 and later.

Declarative Metadata Sample Definition

The following is an example of the package file.

```
<?xml version="1.0" encoding="UTF-8"?>
<Package xmlns="http://soap.sforce.com/2006/04/metadata">
    <types>
        <members>Quote</members>
        <name>Settings</name>
    </types>
    <version>28.0</version>
</Package>
```

The package file references the following Quote.settings file.

```
<?xml version="1.0" encoding="UTF-8"?>
<QuoteSettings xmlns="http://soap.sforce.com/2006/04/metadata">
    <enableQuote>true</enableQuote>
    <enableQuotesWithoutOppEnabled>true</enableQuotesWithoutOppEnabled>
</QuoteSettings>
```

Wildcard Support in the Manifest File

The wildcard character `*` (asterisk) in the `package.xml` manifest file doesn't apply to metadata types for feature settings. The wildcard applies only when retrieving all settings, not for an individual setting. For details, see Settings. For information about using the manifest file, see Deploying and Retrieving Metadata with the Zip File.

RecordPageSettings

Represents an org's record page settings. This type extends the Metadata metadata type and inherits its `fullName` field.

In the package manifest, all organization settings metadata types are accessed using the "Settings" name. See Settings for more details.

Declarative Metadata File Suffix and Directory Location

RecordPageSettings values are stored in a single file named `RecordPage.settings` in the `settings` directory. The `.settings` files are different from other named components because there is only one settings file for each settings component.

Version

Record page settings are available in API version 47.0 and later.

Fields

Field	Field Type	Description
`enableActivityRelatedList`	boolean	Indicates whether the default activities view is related lists (`true`) or activity timeline (`false`).
`enableFullRecordView`	boolean	Indicates whether the default record page view is full view (`true`) or grouped view (`false`).

Declarative Metadata Sample Definition

This is a sample `recordpage.settings` metadata file.

```
<?xml version="1.0" encoding="UTF-8"?>
<RecordPageSettings xmlns=""http://soap.sforce.com/2006/04/metadata"">
    <enableActivityRelatedList>true</enableActivityRelatedList>
    <enableFullRecordView>true</enableFullRecordView>
</RecordPageSettings>
```

Example Package Manifest

The following is an example package manifest used to deploy or retrieve the Record Page settings metadata for an organization

```
<?xml version="1.0" encoding="UTF-8"?>
<Package xmlns=""http://soap.sforce.com/2006/04/metadata"">
    <types>
        <members>RecordPage</members>
        <name>Settings</name>
    </types>
    <version>47.0</version>
</Package>
```

Wildcard Support in the Manifest File

The wildcard character `*` (asterisk) in the `package.xml` manifest file doesn't apply to metadata types for feature settings. The wildcard applies only when retrieving all settings, not for an individual setting. For details, see Settings. For information about using the manifest file, see Deploying and Retrieving Metadata with the Zip File.

SEE ALSO:

Settings

SchemaSettings

Represents an org's schema settings, which manage the availability of custom settings and custom metadata type values. This type extends the Metadata metadata type and inherits its `fullName` field.

In the package manifest, all organization settings metadata types are accessed using the "Settings" name. See Settings for more details.

File Suffix and Directory Location

SchemaSettings values are stored in the `Schema.settings` file in the `settings` directory. The `.settings` files are different from other named components because there is only one settings file for each settings component.

Version

SchemaSettings is available in API version 47.0 and later.

Fields

Field Name	Field Type	Description
`enableAdvancedCMTSecurity`	boolean	Indicates whether custom metadata type values are available only to Apex, flow, and formula operations (`true`) or exposed in other contexts such as through the Enterprise WSDL or SOAP API (`false`). This field has a default value of `false`.
`enableAdvancedCSSecurity`	boolean	Indicates whether custom settings type values are available only to Apex, flow, and formula operations (`true`) or exposed in other contexts such

Field Name	Field Type	Description
		as through the Enterprise WSDL or SOAP API (`false`). This field has a default value of `false`.
`enableListCustomSettingCreation`	boolean	Indicates whether you can create custom settings when using application-level data definitions (`true`) or not (`false`). This field has a default value of `false`.
`enableSOSLOnCustomSettings`	boolean	Indicates whether custom settings values are returned in Salesforce Object Search language (SOSL) queries (`true`) or not (`false`). This field has a default value of `false`.

Declarative Metadata Sample Definition

The following is an example of a SchemaSettings component.

```
<?xml version="1.0" encoding="UTF-8"?>
<SchemaSettings xmlns="http://soap.sforce.com/2006/04/metadata">
    <enableAdvancedCMTSecurity>true</enableAdvancedCMTSecurity>
    <enableAdvancedCSSecurity>true</enableAdvancedCSSecurity>
    <enableListCustomSettingCreation>false</enableListCustomSettingCreation>
    <enableSOSLOnCustomSettings>true</enableSOSLOnCustomSettings>
</SchemaSettings>
```

The following is an example `package.xml` that references the previous definition.

```
<?xml version="1.0" encoding="UTF-8"?>
<Package xmlns="http://soap.sforce.com/2006/04/metadata">
    <types>
        <members>Schema</members>
        <name>Settings</name>
    </types>
    <version>47.0</version>
</Package>
```

Wildcard Support in the Manifest File

The wildcard character `*` (asterisk) in the `package.xml` manifest file doesn't apply to metadata types for feature settings. The wildcard applies only when retrieving all settings, not for an individual setting. For details, see Settings. For information about using the manifest file, see Deploying and Retrieving Metadata with the Zip File.

SearchSettings

Represents an org's search settings. This type extends the Metadata metadata type and inherits its `fullName` field.

In the package manifest, all organization settings metadata types are accessed using the "Settings" name. See Settings for more details.

File Suffix and Directory Location

SearchSettings values are stored in a single file named `Search.settings` in the `settings` folder. The `.settings` files are different from other named components because there is only one settings file for each settings component.

Version

SearchSettings is available in API version 37.0 and later.

Fields

Field Name	Field Type	Description
`documentContentSearchEnabled`	boolean	Indicates whether a full-text document search is performed.
`enableAdvancedSearchInAlohaSidebar`	boolean	Indicates whether advanced search is available in the search sidebar (`true`) or not (`false`). Available in Salesforce Classic only. Available in API version 46.0 and later.
`enableEinsteinSearchPersonalization`	boolean	Indicates whether search personalization is enabled (`true`) or not (`false`). Available in Lightning Experience only. Available in API version 47.0 and later.
`enablePersonalTagging`	boolean	Indicates whether users are allowed to group records from various objects by a common theme (`true`) or not (`false`). Personal tags are visible to the user only. Available in Salesforce Classic only. Available in API version 48.0 and later.
`enablePublicTagging`	boolean	Indicates whether users are allowed to group records from various objects by a common theme (`true`) or not (`false`). Personal tags are visible to all users. Available in Salesforce Classic only. Available in API version 48.0 and later.
`enableSalesforceGeneratedSynonyms`	boolean	Indicates whether search synonyms are enabled (`true`) or not (`false`). Available in API version 47.0 and later.
`enableSearchTermHistory`	boolean	Indicates whether users are allowed to group records from various objects by a common theme (`true`) or not (`false`). Public tags are visible to everyone in the organization. Available in Salesforce Classic only. Available in API version 48.0 and later.
`enableSetupSearch`	boolean	Indicates whether the search box in the Setup sidebar returns matching custom fields, custom objects, and other supported setup items when you press Enter (`true`) or not (`false`). The default is `true` in Developer, Performance, Professional, Enterprise, and Unlimited editions, and `false` in all other editions. Available in API version 47.0 and later.
`enableSuggestArticlesLinksOnly`	boolean	Indicates whether links are provided to knowledge articles from Cases similar to the current Case (`true`) or not (`false`). Available in API version 48.0 and later.
`enableUseDefaultSearchEntity`	boolean	Indicates whether to use the admin-specified default entity in sidebar search (`true`) or not (`false`). Available in Salesforce Classic only. Available in API version 48.0 and later.

Field Name	Field Type	Description
`optimizeSearchForCJKEnabled`	boolean	Required. Indicates whether the search is optimized for the Japanese, Chinese, and Korean languages (`true`) or not (`false`). This setting affects sidebar search and the account search for **Find Duplicates** on a lead record in sidebar search and global search. Enable this option if users are searching mostly in Japanese, Chinese, or Korean, and if the text in searchable fields is mostly in those languages.
`recentlyViewedUsersForBlankLookupEnabled`	boolean	Required. Indicates whether the list of records that are returned from a user autocomplete lookup and from a blank user lookup is taken from the user's recently viewed user records (`true`). Otherwise this setting is `false` if the lookup shows a list of recently accessed user records from across your org (`false`). Only applies to User object blank lookup searches.
`searchSettingsByObject`	SearchSettingsByObject	Required. Represents a list of search settings for each object.
`sidebarAutoCompleteEnabled`	boolean	Required. Indicates whether autocomplete is enabled for sidebar search (`true`) or not (`false`). Autocomplete is when users start typing search terms and sidebar search displays a matching list of recently viewed records.
`sidebarDropDownListEnabled`	boolean	Required. Indicates whether a dropdown list appears in the sidebar search section (`true`) or not (`false`). From this list, users can select to search within tags, within a specific object, or across all objects.
`sidebarLimitToItemsIOwnCheckboxEnabled`	boolean	Required. Indicates whether the **Limit to Items I Own** checkbox appears (`true`) or not (`false`). The checkbox allows your users to include only records for which they are the record owner when entering search queries in the sidebar.
`singleSearchResultShortcutEnabled`	boolean	Required. Indicates whether a shortcut is enabled (`true`) or not (`false`). With the shortcut, users skip the search results page and go directly to the record's detail page when their search returns only a single item. This setting doesn't apply to tags, case comments (in advanced search), and global search.
`spellCorrectKnowledgeSearchEnabled`	boolean	Required. Indicates whether spell check is enabled for Knowledge search (`true`) or not (`false`).

SearchSettingsByObject

Field Name	Field Type	Description
`searchSettingsByObject`	ObjectSearchSetting	Contains a list of search settings for each object.

ObjectSearchSetting

A list of search settings for each object.

Field Name	Field Type	Description
`enhancedLookupEnabled`	boolean	Required. Indicates whether enhanced lookups is enabled for the object (`true`) or not (`false`).
`lookupAutoCompleteEnabled`	boolean	Required. Indicates whether autocomplete is enabled for lookup search (`true`) or not (`false`). Autocomplete is when users edit the lookup field inline by choosing an autosuggestion.
`name`	string	Required. The entity name of the object being configured.
`resultsPerPageCount`	int	Required. The number of search results per page.

Declarative Metadata Sample Definition

The following is an example of the `Search.settings` file.

```
<?xml version="1.0" encoding="UTF-8"?>
    <SearchSettings xmlns="http://soap.sforce.com/2006/04/metadata">
        <enableSetupSearch>false</enableSetupSearch>
        <enableAdvancedSearchInAlohaSidebar>false</enableAdvancedSearchInAlohaSidebar>
        <enableQuerySuggestionPigOn>false</enableQuerySuggestionPigOn>
        <enableSalesforceGeneratedSynonyms>false</enableSalesforceGeneratedSynonyms>
        <enableSearchTermHistory>false</enableSearchTermHistory>
        <enablePublicTagging>false</enablePublicTagging>
        <enablePersonalTagging>false</enablePersonalTagging>
        <enableSuggestArticlesLinksOnly>false</enableSuggestArticlesLinksOnly>
        <enableUseDefaultSearchEntity>false</enableUseDefaultSearchEntity>
            <documentContentSearchEnabled>true</documentContentSearchEnabled>
            <optimizeSearchForCJKEnabled>true</optimizeSearchForCJKEnabled>

<recentlyViewedUsersForBlankLookupEnabled>true</recentlyViewedUsersForBlankLookupEnabled>

            <searchSettingsByObject>
            <searchSettingsByObject>
                <enhancedLookupEnabled>false</enhancedLookupEnabled>
                <lookupAutoCompleteEnabled>false</lookupAutoCompleteEnabled>
                <name>Account</name>
                <resultsPerPageCount>25</resultsPerPageCount>
            </searchSettingsByObject>
            <searchSettingsByObject>
                <enhancedLookupEnabled>false</enhancedLookupEnabled>
                <lookupAutoCompleteEnabled>false</lookupAutoCompleteEnabled>
                <name>Activity</name>
                <resultsPerPageCount>25</resultsPerPageCount>
            </searchSettingsByObject>
            <searchSettingsByObject>
                <enhancedLookupEnabled>false</enhancedLookupEnabled>
                <lookupAutoCompleteEnabled>false</lookupAutoCompleteEnabled>
                <name>Asset</name>
                <resultsPerPageCount>25</resultsPerPageCount>
            </searchSettingsByObject>
            </searchSettingsByObject>
            <sidebarAutoCompleteEnabled>true</sidebarAutoCompleteEnabled>
            <sidebarDropDownListEnabled>true</sidebarDropDownListEnabled>

<sidebarLimitToItemsIOwnCheckboxEnabled>true</sidebarLimitToItemsIOwnCheckboxEnabled>
            <singleSearchResultShortcutEnabled>true</singleSearchResultShortcutEnabled>
            <spellCorrectKnowledgeSearchEnabled>true</spellCorrectKnowledgeSearchEnabled>

        <enableEinsteinSearchPersonalization>true</enableEinsteinSearchPersonalization>
    </SearchSettings>
```

Example Package Manifest

The following is an example package manifest used to deploy or retrieve the Search settings metadata for an organization.

```
<?xml version="1.0" encoding="UTF-8"?>
<Package xmlns="http://soap.sforce.com/2006/04/metadata">
    <types>
        <members>Search</members>
        <name>Settings</name>
    </types>
    <version>37.0</version>
</Package>
```

Wildcard Support in the Manifest File

The wildcard character `*` (asterisk) in the `package.xml` manifest file doesn't apply to metadata types for feature settings. The wildcard applies only when retrieving all settings, not for an individual setting. For details, see Settings. For information about using the manifest file, see Deploying and Retrieving Metadata with the Zip File.

SecuritySettings

Represents an org's security settings. For example, settings define trusted IP ranges for network access, password and login requirements, session expiration, and single sign-on settings. This type extends the Metadata metadata type and inherits its `fullName` field.

In the package manifest, all organization settings metadata types are accessed using the "Settings" name. See Settings for more details.

File Suffix and Directory Location

SecuritySettings values are stored in a single file named `Security.settings` in the `settings` directory. The `.settings` files are different from other named components because there is only one settings file for each settings component.

Version

Security settings are available in API version 27.0 and later. API versions 26 and earlier are no longer available.

Fields

Field Name	Field Type	Description
`canUsersGrantLoginAccess`	boolean	If `true`, users can grant login access to Support. If `false`, only an admin can grant login access. Note: Users can't grant login access to managed packages that are licensed to your entire Salesforce org. Only admins with the "Manage Users" permission enabled on their profiles can grant access to these publishers. Also, some managed packages don't have login access. If a package isn't listed on the Login Access Policies page, login access isn't available for that package.
`enableAdminLoginAsAnyUser`	boolean	If `true`, the **Administrators Can Log in as Any User** field is enabled. The default is not enabled (`false`).
`enableAuditFieldsInactiveOwner`	boolean	If `true`, this setting enables audit fields and updating the owner for records that are owned by inactive users. The default value is `false`. This field is available in API version 47.0 and later.
`enableAuraSecureEvalPref`	boolean	If `true`, this setting prevents the creation of function expressions in dynamically created Aura components. The default is `false`. This field is available in API version 47.0 and later.
`enableRequireHttpsConnection`	boolean	Deprecated in API version 47.0 and later.
`isTLSv12Required`	boolean	Indicates whether connections to or from your Salesforce org must use TLS 1.2 or higher (`true`) or not (`false`). This field has a default value of `false`. Available in API version 47.0 and later.
`isTLSv12RequiredCommunities`	boolean	Indicates whether connections to or from your Salesforce communities, sites, and portals must use TLS 1.2 or higher (`true`) or not (`false`). This field has a default value of `false`. Available in API version 47.0 and later.
`networkAccess`	NetworkAccess	The trusted IP address ranges from which users can always log in without requiring computer activation.
`passwordPolicies`	PasswordPolicies	The requirements for passwords and logins, and assistance with retrieving forgotten passwords.
`sessionSettings`	SessionSettings	The settings for session expiration and security.
`singleSignOnSettings`	SingleSignOnSettings	The settings for single sign-on (SSO).

NetworkAccess

Represents your org's trusted IP address ranges for network access.

Field	Field Type	Description
`ipRanges`	IpRange[]	The trusted IP address ranges from which users can always log in without requiring computer activation. Note: To add an IP range, deploy all existing IP ranges, including the one you want to add. Otherwise, the existing IP ranges are replaced with the ones you deploy. To remove all the IP ranges, leave the networkAccess field blank (<networkAccess></networkAccess>).

IpRange

Defines a range of trusted IP addresses for network access.

Field	Field Type	Description
`description`	string	The description of the trusted IP range. Use this field to identify the range, such as which corporate network corresponds to this range. Available in API version 34.0 and later.
`end`	string	The IP address that defines the high end of a range of trusted addresses.
`start`	string	The IP address that defines the low end of a range of trusted addresses.

PasswordPolicies

Represents your org’s password and login policies, which show up under **Security Controls | Password Policies**.

Field	Field Type	Description
`apiOnlyUserHomePageURL`	string	The URL to which users with the “API Only User” permission are redirected instead of the login page.
`complexity`	Complexity (enumeration of type string)	The types of characters that must be used in a user’s password. Valid values are: • `NoRestriction`—Has no requirements and is the least secure option. • `AlphaNumeric`—The default setting. Requires at least one alphabetic character and one number. This value is the default value. • `SpecialCharacters`—Requires at least one alphabetic character, one number, and one of the following characters: `! " # $ % & ' ( ) * + , - . / : ; < = > ? @ [ \ ] ^ _ ` { | } ~`.

Field	Field Type	Description
		- `UpperLowerCaseNumeric`—Requires at least one number, one uppercase letter, and one lowercase letter. This value is available in API version 31.0 and later. - `UpperLowerCaseNumericSpecialCharacters`—Requires at least one number, one uppercase letter, one lowercase letter, and one of the following characters: `! " # $ % & ' ( ) * + , - . / : ; < = > ? @ [ \ ] ^ _ ` { | } ~`. This value is available in API version 31.0 and later. - `Any3UpperLowerCaseNumericSpecialCharacters`—Requires at least three of the following options: one number, one uppercase letter, one lowercase letter, and one special character (`! " # $ % & ' ( ) * + , - . / : ; < = > ? @ [\ ] ^ _ ` { | } ~`). This value is available in API version 46.0 and later.
`enableSetPasswordInApi`	boolean	If `true`, you can use the `setPassword()` API to change a password without providing the old password. Available in API version 47.0 and later.
`expiration`	Expiration (enumeration of type string)	The length of time until a user password expires and must be changed. Valid values are: - `Never` - `ThirtyDays` - `SixtyDays` - `NinetyDays`. This value is the default value. - `SixMonths` - `OneYear`
`historyRestriction`	string	The number of previous passwords saved for users so that they must always reset a new, unique password. Valid values are `0` through `24` passwords remembered. The maximum value of 24 applies to API version 31.0 and later. In earlier versions, the maximum value is 16. The default value is `3`.
`lockoutInterval`	LockoutInterval (enumeration of type string)	The duration of the login lockout. Valid values are: - `FifteenMinutes`. This value is the default value. - `ThirtyMinutes` - `SixtyMinutes` - `Forever` (must be reset by admin)
`maxLoginAttempts`	MaxLoginAttempts (enumeration of type string)	The number of login failures allowed for a user before the user is locked out. Valid values are: - `NoLimit`

Field	Field Type	Description
		- `ThreeAttempts` - `FiveAttempts` - `TenAttempts`. This value is the default value.
`minimumPasswordLength`	string	The minimum number of characters required for a password. The number can contain from 5 to 50 characters (default is 8). Available in API version 35.0 and later. Before API version 35.0, specify minimum password length with the enumeration `minPasswordLength`, with valid values `FiveCharacters`, `EightCharacters` (default), `TenCharacters`, `TwelveCharacters` (API version 31.0 and later), and `FifteenCharacters` (API version 34.0 and later).
`minimumPasswordLifetime`	boolean	If **Require a minimum 1 day password lifetime** is enabled (`true`), passwords can't be changed more than once during a 24-hour period. The default is `false`. Available in API version 31.0 and later.
`obscureSecretAnswer`	boolean	If enabled (`true`), hide answers to security questions as the user types. The default is `false`. Note: If your org uses the Microsoft Input Method Editor (IME) with the input mode set to Hiragana, when you type ASCII characters, they're converted in to Japanese characters in normal text fields. However, the IME doesn't work properly in fields with obscured text. If your org's users cannot properly enter their passwords or other values after enabling this feature, disable the feature.
`passwordAssistanceMessage`	string	The text that appears in the Account Lockout email and at the bottom of the Confirm Identity screen for users resetting their passwords.
`passwordAssistanceURL`	string	The URL that users can click to retrieve forgotten passwords.
`questionRestriction`	QuestionRestriction (enumeration of type string)	The restriction on whether the answer to the password hint question can contain the password itself. Valid values are: - `None` - `DoesNotContainPassword`. This value is the default value.

SessionSettings

Represents your org's session expiration and security settings.

Field	Field Type	Description
`allowUserAuthenticationByCertificate`	boolean	If enabled (`true`), users can authenticate with a PEM-encoded X.509 digital certificate. Not enabled by default. Available in API version 47.0 and later.
`canConfirmEmailChangeInLightningCommunities`	boolean	If **Require email confirmations for email address changes** is enabled (`true`), when users change their email address, they receive an email at the new address with a link. After they click the link, their new email address takes effect. For orgs created before Winter '20, the field is not enabled by default. For new orgs, this field is always enabled. To disable the field (not recommended), contact Salesforce Customer Support. Available in API version 47.0 and later.
`canConfirmIdentityBySmsOnly`	boolean	Prevents identity verification by email for users who have registered other verification methods, such as SMS or Salesforce Authenticator. If no other verification methods are configured, users are verified by email. By default, this setting is disabled (`false`) for existing orgs. For new orgs, this setting is enabled (`true`) by default. Available in API version 48.0 and later.
`disableTimeoutWarning`	boolean	Indicates whether the session timeout warning popup is disabled (`true`) or enabled (`false`).
`enableCSPOnEmail`	boolean	Indicates whether a content security policy is enabled for the email template. A content security policy helps prevent cross-site scripting attacks by whitelisting sources of images and other content.
`enableCSRFOnGet`	boolean	Indicates whether Cross-Site Request Forgery (CSRF) protection on GET requests on non-setup pages is enabled (`true`) or disabled (`false`).
`enableCSRFOnPost`	boolean	Indicates whether Cross-Site Request Forgery (CSRF) protection on POST requests on non-setup pages is enabled (`true`) or disabled (`false`).
`enableCacheAndAutocomplete`	boolean	Indicates whether the user's browser is allowed to store usernames and auto-fill the `User Name` field on the login page (`true`) or not (`false`).
`enableClickjackNonsetupSFDC`	boolean	Indicates whether clickjack protection for non-setup Salesforce pages is enabled (`true`) or disabled (`false`).
`enableClickjackNonsetupUser`	boolean	Indicates whether clickjack protection for customer Visualforce pages with standard headers turned on is enabled (`true`) or disabled (`false`).
`enableClickjackNonsetupUserHeaderless`	boolean	Indicates whether clickjack protection for customer Visualforce pages with standard headers turned off is enabled (`true`) or disabled (`false`). Available in API version 34.0 and later.

Field	Field Type	Description
`enableClickjackSetup`	boolean	Indicates whether clickjack protection for setup pages is enabled (`true`) or disabled (`false`).
`enableContentSniffingProtection`	boolean	Indicates if the browser is prevented from inferring the MIME type from the document content and from executing malicious files (JavaScript, Stylesheet) as dynamic content. This field is available in API version 39.0 and later.
`enableLightningLogin`	boolean	If enabled (`true`), users can use Lightning Login (Salesforce Authenticator) to log in instead of a password. Available in API Version 47.0 and later.
`enableLightningLoginOnlyWithUserPerm`	boolean	If enabled (`true`), only users with the Lightning Login User permission can log in with Salesforce Authenticator instead of a password. Available in API Version 47.0 and later.
`enablePostForSessions`	boolean	Indicates whether cross-domain session information is exchanged using a POST request instead of a GET request, such as when a user is using a Visualforce page. In this context, POST requests are more secure than GET requests. Available in API version 31.0 and later.
`enableSMSIdentity`	boolean	If enabled (`true`), the default, users can receive a one-time password in a text message (SMS) to verify their identity. Users must verify their mobile phone number before they can receive SMS messages.
`enableU2F`	boolean	If enabled (`true`), users can use a physical device (U2F) for two-factor authentication and identity verification. The default is `false`. Available in API version 47.0 and later.
`enableUpgradeInsecureRequests`	boolean	Indicates if HTTPS is required for connecting to third-party domains. Note: This setting is enabled by default on accounts created after the Summer '17 release. This field is available in API version 42.0 and later.
`enableXssProtection`	boolean	Indicates if protection against reflected cross-site scripting attacks is enabled. If a reflected cross-site scripting attack is detected and XSS protection is enabled, the browser shows a blank page with no content. This field is available in API version 39.0 and later.
`enforceIpRangesEveryRequest`	boolean	If `true`, the IP addresses in Login IP Ranges are enforced when a user accesses Salesforce (on every page request), including access from a client app. If `false`, the IP addresses in Login IP Ranges are enforced only when a user logs in. This field affects

Field	Field Type	Description
		all user profiles that have login IP restrictions. Available in API version 34.0 and later.
`forceLogoutOnSessionTimeout`	boolean	If enabled (`true`), the default, when sessions time out for inactive users, current sessions become invalid. The browser refreshes and returns to the login page. To access the org, the user must log in again. Available in API version 31.0 and later.
`forceRelogin`	boolean	If `true`, an admin who is logged in as another user must log in again to their original session, after logging out as the secondary user. If `false`, the admin is not required to log in again.
`hasRetainedLoginHints`	boolean	If you enable **Remember me until logout** (`true`), usernames (login hints) are cached until the user logs out. If a session times out, usernames appear on the Switcher as inactive. If `false` (default), usernames aren't cached for SSO sessions.
`hasUserSwitching`	boolean	If **Enable user switching** is`true` (default), when users select their profile picture, the Switcher shows up, which lets users log in to other orgs. The **Enable caching and autocomplete on login page** setting must also be enabled. If `false`, the Switcher doesn't show up when users select their profile picture, and your org doesn't appear in Switchers on other orgs.
`hstsOnForcecomSites`	boolean	Indicates whether Visualforce, Salesforce sites, or Communities must use HTTPS. Available in API version 41.0 and later.
`identityConfirmationOnEmailChange`	boolean	Indicates if a user's identity is confirmed when changing their email address, instead of requiring a relogin. This field is available in API version 42.0 and later.
`identityConfirmationOnTwoFactorRegistrationEnabled`	boolean	Indicates if users are required to confirm their identities to add a two-factor authentication method, such as Salesforce Authenticator, instead of requiring a relogin. This field is available in API version 40.0 and later.
`lockSessionsToDomain`	boolean	Indicates whether the current UI session for a user, such as a community user, is associated with a specific domain. This check helps prevent unauthorized use of the session ID in another domain. The value is `true` by default for orgs created with the Spring '15 release or later. Available in API version 33.0 and later.
`lockSessionsToIp`	boolean	Indicates whether user sessions are locked to the IP address from which the user logged in (`true`) or not (`false`).
`logoutURL`	string	The URL to which users are redirected when they log out of Salesforce. If no value is specified, the default is

Field	Field Type	Description
		`https://login.salesforce.com` unless MyDomain is enabled. If My Domain is enabled, the default is `https://customdomain.my.salesforce.com`. Available in API version 34.0 and later.
`redirectionWarning`	boolean	Indicates whether users see an alert when they click a link in a web tab that redirects them outside the salesforce.com domain. Available in API version 42.0 and later.
`referrerPolicy`	boolean	Indicates whether the referrer header hides sensitive information that could be present in the full URL. If `true`, then the referrer header displays only `salesforce.com`. If `false`, then the header displays the entire URL. For a Visualforce user, if `referrerPolicy` is set to `true`, then the referrer header displays only `force.com`. If `false`, then the header displays the entire URL. Available in API version 42.0 and later. Note: When a request is made from a domain within salesforce.com or force.com, the referrer header displays the entire URL.
`requireHttpOnly`	boolean	Sets the `HttpOnly` attribute on session cookies, making them inaccessible via JavaScript. If `true`, session ID cookie access is restricted. If `false`, access is not restricted. Note: If you have a custom or packaged application that uses JavaScript to access session ID cookies, your application breaks if `requireHttpOnly` is set to `true`. The application can't access the cookie. This field is available in API version 40.0 and later.
`requireHttps`	boolean	Determines whether HTTPS is required to log in to or access Salesforce. This option is enabled by default for security reasons and can't be disabled. To change to HTTP, contact Salesforce Customer Support. This field is available in API version 40.0 and later.
`securityCentralKillSession`	boolean	Deprecated in API version 36.0 and later.
`sessionTimeout`	SessionTimeout (enumeration of type string)	The length of time after which users without activity are prompted to log out or continue working. Valid values are: • `FifteenMinutes` • `ThirtyMinutes` • `SixtyMinutes` • `TwoHours` • `FourHours` • `EightHours`

Field	Field Type	Description
		• `TwelveHours`

SingleSignOnSettings

Represents your org's single sign-on (SSO) settings. These settings are available API version 47.0 and later.

Field Name	Field Type	Description
`enableForceDelegatedCallout`	boolean	If you enable **Force Delegated Authentication Callout** (`true`), a callout to the SSO endpoint occurs regardless of login restriction failures. If disabled (`false`), the default, and if a user's first login attempt fails due to login restrictions within the Salesforce org, a call isn't made to the SSO endpoint.
`enableMultipleSamlConfigs`	boolean	If `true` (default), you can configure multiple SAML providers. Once enabled, the setting can't be disabled.
`enableSamlJitProvisioning`	boolean	If you enable **User Provisioning Enabled** (`true`), you can provision users through a SAML assertion (called just-in-time provisioning). Requires `EnableSamlLogin` to be `true` and `enableMultipleSamlConfigs` to be `false`. The default is not enabled (`false`).
`enableSamlLogin`	boolean	If you enable **SAML Enabled** (`true`), users can SSO into Salesforce from providers via SAML. The default is not enabled (`false`).

Declarative Metadata Sample Definition

The following is a sample `security.settings` metadata file.

```
<?xml version="1.0" encoding="UTF-8"?>
<SecuritySettings xmlns="http://soap.sforce.com/2006/04/metadata"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
    <enableAdminLoginAsAnyUser xsi:nil="true"/>
    <enableAuraSecureEvalPref xsi:nil="true"/>
    <enableAuditFieldsInactiveOwner xsi:nil="true"/>
    <enableSetRequiredHttpsConnection xsi:nil="true"/>
    <enableTurnTLSv11Off xsi:nil="true"/>
    <enableTurnTLSv11OffSites xsi:nil="true"/>
    <enableTurnTLSv1Off xsi:nil="true"/>
    <networkAccess/>
    <passwordPolicies>
        <complexity>NoRestriction</complexity>
        <enableSetPasswordInApi>false</enableSetPasswordInApi>
        <expiration>Never</expiration>
        <historyRestriction>0</historyRestriction>
        <lockoutInterval>FifteenMinutes</lockoutInterval>
        <maxLoginAttempts>TenAttempts</maxLoginAttempts>
        <minimumPasswordLength>5</minimumPasswordLength>
        <minimumPasswordLifetime>false</minimumPasswordLifetime>
        <obscureSecretAnswer>false</obscureSecretAnswer>
        <questionRestriction>DoesNotContainPassword</questionRestriction>
    </passwordPolicies>
    <sessionSettings>
       <allowUserAuthenticationByCertificate>false</allowUserAuthenticationByCertificate>

        <disableTimeoutWarning>false</disableTimeoutWarning>
        <enableCSPOnEmail>true</enableCSPOnEmail>
        <enableCSRFOnGet>true</enableCSRFOnGet>
        <enableCSRFOnPost>true</enableCSRFOnPost>
        <enableCacheAndAutocomplete>true</enableCacheAndAutocomplete>
        <enableClickjackNonsetupSFDC>true</enableClickjackNonsetupSFDC>
        <enableClickjackNonsetupUser>false</enableClickjackNonsetupUser>
       <enableClickjackNonsetupUserHeaderless>false</enableClickjackNonsetupUserHeaderless>

        <enableClickjackSetup>true</enableClickjackSetup>
        <enableContentSniffingProtection>true</enableContentSniffingProtection>
        <enableLightningLogin>true</enableLightningLogin>
       <enableLightningLoginOnlyWithUserPerm>false</enableLightningLoginOnlyWithUserPerm>

        <enablePostForSessions>false</enablePostForSessions>
        <enableSMSIdentity>true</enableSMSIdentity>
        <enableU2F>false</enableU2F>
        <enableUpgradeInsecureRequests>true</enableUpgradeInsecureRequests>
        <enableXssProtection>true</enableXssProtection>
        <enforceIpRangesEveryRequest>false</enforceIpRangesEveryRequest>
        <forceLogoutOnSessionTimeout>true</forceLogoutOnSessionTimeout>
        <forceRelogin>true</forceRelogin>
        <hasRetainedLoginHints>false</hasRetainedLoginHints>
        <hasUserSwitching>true</hasUserSwitching>
        <hstsOnForcecomSites>false</hstsOnForcecomSites>
        <identityConfirmationOnEmailChange>true</identityConfirmationOnEmailChange>

<identityConfirmationOnTwoFactorRegistrationEnabled>true</identityConfirmationOnTwoFactorRegistrationEnabled>

        <lockSessionsToDomain>true</lockSessionsToDomain>
        <lockSessionsToIp>false</lockSessionsToIp>
        <lockerServiceCSP>true</lockerServiceCSP>
        <redirectionWarning>true</redirectionWarning>
        <referrerPolicy>true</referrerPolicy>
        <requireHttpOnly>false</requireHttpOnly>
        <requireHttps>false</requireHttps>
        <sessionTimeout>TwoHours</sessionTimeout>
    </sessionSettings>
    <singleSignOnSettings>
        <enableForceDelegatedCallout>false</enableForceDelegatedCallout>
        <enableMultipleSamlConfigs>true</enableMultipleSamlConfigs>
        <enableSamlJitProvisioning>false</enableSamlJitProvisioning>
        <enableSamlLogin>false</enableSamlLogin>
    </singleSignOnSettings>
</SecuritySettings>
```

The following is an example `package.xml` manifest that references the previous definition.

```
?xml version="1.0" encoding="UTF-8"?>
<Package xmlns="http://soap.sforce.com/2006/04/metadata">
    <types>
        <members>Security</members>
        <name>Settings</name>
    </types>
    <version>47.0</version>
</Package>
```

Wildcard Support in the Manifest File

The wildcard character `*` (asterisk) in the `package.xml` manifest file doesn't apply to metadata types for feature settings. The wildcard applies only when retrieving all settings, not for an individual setting. For details, see Settings. For information about using the manifest file, see Deploying and Retrieving Metadata with the Zip File.

SharingSettings

Represents an organization's sharing, visibility, and data access settings. This type extends the Metadata metadata type and inherits its `fullName` field.

In the package manifest, all organization settings metadata types are accessed using the "Settings" name. See Settings for more details.

File Suffix and Directory Location

SharingSettings values are stored in the `Sharing.settings` file in the `settings` directory. The `.settings` files are different from other named components because there is only one settings file for each settings component.

Version

SharingSettings is available in API version 47.0 and later.

Special Access Rules

To use SharingSettings, you need the Manage Sharing permission.

Fields

Field Name	Field Type	Description
`enableAccountRoleOptimization`	boolean	Indicates whether person roles are assigned to new community users in accounts without existing users (`true`) or if portal roles are created for new users (`false`). This field has a default value of `false`.
`enableAssetSharing`	boolean	Indicates whether sharing is enabled for assets (`true`) or asset access is determined by the parent object's sharing rules (`false`). This field has a default value of `false`.

Field Name	Field Type	Description
enableCommunityUserVisibility	boolean	Indicates whether community users in the same community can see each other regardless of the organization-wide defaults (true) or not (false). This field has a default value of false. In orgs created in API version 47.0 and later, this setting doesn't apply to guest users.
enableExternalSharingModel	boolean	Indicates whether the external sharing model is enabled (true) or not (false). This field has a default value of true if communities are enabled, and a default value of false if not. To use this field, you need the Customize Application permission.
enableManagerGroups	boolean	Indicates whether users can share records with their managers and manager subordinates groups (true) or not (false). This field has a default value of false. To use this field, you need the View and Manage Users permission.
enableManualUserRecordSharing	boolean	Indicates whether users can share their own user record (true) or not (false). This field has a default value of false.
enablePartnerSuperUserAccess	boolean	Indicates whether you can grant super user access to partners in communities (true) or not (false). This field has a default value of false. To use this field, you need the Customize Application permission
enablePortalUserCaseSharing	boolean	Indicates whether portal users can access related contacts for cases that they own (true) or not (false). This field has a default value of false.
enablePortalUserVisibility	boolean	Indicates whether portal users in the same customer or partner portal account can see each other regardless of the organization-wide defaults (true) or not (false). This field has a default value of false. To enable this field, contact Salesforce Support.
enableRemoveTMGroupMembership	boolean	Removes group membership info for the original territory management feature after migrating to Enterprise Territory Management when set to true. This field has a default value of false. Once this field is set to true, it can't be set to false again.
enableRestrictAccessLookupRecords	boolean	Indicates whether users must have read access to a record to see the record's name in lookup and system fields (true) or not (false). This field has a default value of true in Salesforce orgs created in Spring '20 or later and a default value of false in all other orgs. This field is available in API version 48.0 and later.
enableSecureGuestAccess	boolean	Indicates whether guest users' record access is secured (true) or not (false). If set to true, guest users have org-wide defaults set to Private, and you must use guest user sharing rules to share records with them. This field has a default value of false.
enableStandardReportVisibility	boolean	Indicates whether users can view reports based on standard report types that may expose data of users to whom they don't have access (true) or not (false). This field has a default value of false.

Field Name	Field Type	Description
enableTerritoryForecastManager	boolean	Indicates whether forecast managers can act as delegated administrators for territories below them in the hierarchy (`true`) or not (`false`). This field has a default value of `false`.

Declarative Metadata Sample Definition

The following is an example of a SharingSettings component.

```
<?xml version="1.0" encoding="UTF-8"?>
<SharingSettings xmlns="http://soap.sforce.com/2006/04/metadata">
    <enableAccountRoleOptimization>false</enableAccountRoleOptimization>
    <enableAssetSharing>false</enableAssetSharing>
    <enableCommunityUserVisibility>false</enableCommunityUserVisibility>
    <enableExternalSharingModel>true</enableExternalSharingModel>
    <enableManagerGroups>false</enableManagerGroups>
    <enableManualUserRecordSharing>true</enableManualUserRecordSharing>
    <enablePartnerSuperUserAccess>false</enablePartnerSuperUserAccess>
    <enablePortalUserCaseSharing>false</enablePortalUserCaseSharing>
    <enablePortalUserVisibility>true</enablePortalUserVisibility>
    <enableRemoveTMGroupMembership>false</enableRemoveTMGroupMembership>
    <enableRestrictAccessLookupRecords>true</enableRestrictAccessLookupRecords>
    <enableSecureGuestAccess>false</enableSecureGuestAccess>
    <enableStandardReportVisibility>false</enableStandardReportVisibility>
    <enableTerritoryForecastManager>false</enableTerritoryForecastManager>
</SharingSettings>
```

The following is an example `package.xml` that references the previous definition.

```
<?xml version="1.0" encoding="UTF-8"?>
<Package xmlns="http://soap.sforce.com/2006/04/metadata">
    <types>
        <members>Sharing</members>
        <name>Settings</name>
    </types>
    <version>47.0</version>
</Package>
```

Wildcard Support in the Manifest File

The wildcard character `*` (asterisk) in the `package.xml` manifest file doesn't apply to metadata types for feature settings. The wildcard applies only when retrieving all settings, not for an individual setting. For details, see Settings. For information about using the manifest file, see Deploying and Retrieving Metadata with the Zip File.

SiteSettings

Represents the settings for Salesforce Sites. This type extends the Metadata metadata type and inherits its `fullName` field.

In the package manifest, all organization settings metadata types are accessed using the "Settings" name. See Settings for more details.

File Suffix and Directory Location

SiteSettings values are stored in a single file named `Site.settings` in the `settings` directory. The `.settings` files are different from other named components because there is only one `.settings` file for each settings component.

Version

SiteSettings components are available in API version 47.0 and later.

Fields

Field Name	Field Type	Description
`enableProxyLoginICHeader`	boolean	Indicates whether security tokens for API logins from callouts (in API version 31.0 and earlier) are required (`true`) or not (`false`). The default value is `true`.
`enableSitesRecordReassignOrgPref`	boolean	When `true`, indicates when the org assigns records created by guest users of a site to a default owner in the org. When `false`, the guest user remains the owner of the record. The default value is `false`. Available in API version 48.0 and later.
`enableTopicsInSites`	boolean	Indicates whether guest and authenticated external users can view topics in Salesforce Sites and Salesforce portals (`true`) or not (`false`). The default value is `false`.
`enableWebruntimeBYOTemplate`	boolean	Indicates whether the Build Your Own (LWC) template is available in Experience Builder. The default value is `false`. Available in API version 48.0 and later. Note: The Build Your Own (LWC) template is available as a developer preview in new Developer orgs. The template isn't generally available unless or until Salesforce announces its general availability in documentation or in press releases or public statements. All commands, parameters, and other features are subject to change or deprecation at any time, with or without notice. Don't implement functionality developed with these commands or tools.

Declarative Metadata Sample Definition

The following is an example of a SiteSettings component.

```
<?xml version="1.0" encoding="UTF-8"?>
<SiteSettings xmlns="http://soap.sforce.com/2006/04/metadata">
     <enableProxyLoginICHeader>true</enableProxyLoginICHeader>
     <enableTopicsInSites>false</enableTopicsInSites>
     <enableVisualforceApiAccessAllowed>true</enableVisualforceApiAccessAllowed>
</SiteSettings>
```

The following is an example `package.xml` that references the previous definition.

```
<?xml version="1.0" encoding="UTF-8"?>
<Package xmlns="http://soap.sforce.com/2006/04/metadata">
    <types>
        <members>Site</members>
        <name>Settings</name>
    </types>
    <version>47.0</version>
</Package>
```

Wildcard Support in the Manifest File

The wildcard character `*` (asterisk) in the `package.xml` manifest file doesn't apply to metadata types for feature settings. The wildcard applies only when retrieving all settings, not for an individual setting. For details, see Settings. For information about using the manifest file, see Deploying and Retrieving Metadata with the Zip File.

SocialCustomerServiceSettings

Represents Social Customer Service settings such as how to format inbound content from social posts to cases.This type extends the Metadata metadata type and inherits its `fullName` field.

In the package manifest, all organization settings metadata types are accessed using the "Settings" name. See Settings for more details.

File Suffix and Directory Location

SocialCustomerServiceSettings components have the suffix `settings` and are stored in the `settings` folder. The `.settings` files are different from other named components because there is only one settings file for each settings component.

Version

SocialCustomerServiceSettings is available in API version 41.0 and later.

Fields

Field Name	Field Type	Description
`caseSubjectOption`	CaseSubjectOption (enumeration of type string)	Required. Specifies an option from which inbound social content is formatted to appear in case records' **Case Subject** field. Valid values are: • `SocialPostSource` • `SocialPostContent` • `BuildCustom`
`enableSocialApprovals`	boolean	Indicates whether social approvals are enabled. To learn more, see Enable Social Post Approvals.The default value is `false`. Available in API version 47.0 and later.

Field Name	Field Type	Description
`enableSocialCaseAssignmentRules`	boolean	Indicates whether case assignment rules are enabled. Use case assignment rules to determine how cases are assigned to users or put into queues as they are created. The default value is `false`. Available in API version 47.0 and later.
`enableSocialCustomerService`	boolean	Indicates whether to enable the Social Customer Service feature. The default value is `false`. Available in API version 47.0 and later.
`enableSocialPersonaHistoryTracking`	boolean	Indicates whether to enable Social Persona history tracking. History tracking helps identify who made what changes when, and for differentiating between automatic and manual changes. The default value is `false`. Available in API version 47.0 and later.
`enableSocialPostHistoryTracking`	boolean	Indicates whether to enable Social Post history tracking. History tracking helps identify who made what changes when, and for differentiating between automatic and manual changes. The default value is `false`. Available in API version 47.0 and later.
`enableSocialReceiveParentPost`	boolean	Indicates whether to use the original social post that initiated the case as the parent record. The default value is `false`. Available in API version 47.0 and later.

Declarative Metadata Sample Definition

This is a sample of a `SocialCustomerServiceSettings.settings` file.

```xml
<?xml version="1.0" encoding="UTF-8"?>
<SocialCustomerServiceSettings xmlns="http://soap.sforce.com/2006/04/metadata">
    <caseSubjectOption>SocialPostSource</caseSubjectOption>
    <enableSocialApprovals>true</enableSocialApprovals>
    <enableSocialCaseAssignmentRules>false</enableSocialCaseAssignmentRules>
    <enableSocialCustomerService>true</enableSocialCustomerService>
    <enableSocialPersonaHistoryTracking>false</enableSocialPersonaHistoryTracking>
    <enableSocialPostHistoryTracking>false</enableSocialPostHistoryTracking>
    <enableSocialReceiveParentPost>true</enableSocialReceiveParentPost>
</SocialCustomerServiceSettings>
```

The following is an example `package.xml` that references the previous definition.

```xml
<?xml version="1.0" encoding="UTF-8"?>
<Package xmlns="http://soap.sforce.com/2006/04/metadata">
    <types>
        <members>SocialCustomerService</members>
        <name>Settings</name>
    </types>
    <version>47.0</version>
</Package>
```

Wildcard Support in the Manifest File

The wildcard character `*` (asterisk) in the `package.xml` manifest file doesn't apply to metadata types for feature settings. The wildcard applies only when retrieving all settings, not for an individual setting. For details, see Settings. For information about using the manifest file, see Deploying and Retrieving Metadata with the Zip File.

SocialProfileSettings

Represents org preferences for social media features such as enabling Twitter and Facebook. This type extends the Metadata metadata type and inherits its `fullName` field.

In the package manifest, all organization settings metadata types are accessed using the "Settings" name. See Settings for more details.

File Suffix and Directory Location

SocialProfileSettings values are stored in a single file named `SocialProfile.settings` in the `settings` directory of the corresponding package directory. The `.settings` files are different from other named components because there is only one settings file for each settings component.

Version

SocialProfileSettings is available in API version 47.0 and later.

Fields

Field Name	Field Type	Description
`isFacebookSocialProfilesDisabled`	boolean	Prevents users from accessing Facebook in social CRM (`true`) or not (`false`). `enableSocialProfiles` must be `true` to enable Facebook social profiles.
`isLinkedInSocialProfilesDisabled`	boolean	Prevents users from accessing LinkedIn in social CRM (`true`) or not (`false`). `enableSocialProfiles` must be `true` to enable LinkedIn social profiles.
`isTwitterSocialProfilesDisabled`	boolean	Prevents users from accessing Twitter in social CRM (`true`) or not (`false`). `enableSocialProfiles` must be `true` to enable Twitter social profiles.
`isYouTubeSocialProfilesDisabled`	boolean	Prevents users from accessing YouTube in social CRM (`true`) or not (`false`). `enableSocialProfiles` must be `true` to enable YouTube social profiles.
`enableSocialProfiles`	boolean	Indicates whether users can access social media profiles in social CRM (`true`) or not (`false`).

Wildcard Support in the Manifest File

The wildcard character `*` (asterisk) in the `package.xml` manifest file doesn't apply to metadata types for feature settings. The wildcard applies only when retrieving all settings, not for an individual setting. For details, see Settings. For information about using the manifest file, see Deploying and Retrieving Metadata with the Zip File.

SurveySettings

Represents an org's survey settings. Use the SurveySettings component to enable surveys for your org and choose whether the owner of a survey can manage the responses. This type extends the Metadata metadata type and inherits its `fullName` field.

In the package manifest, all organization settings metadata types are accessed using the "Settings" name. See Settings for more details.

File Suffix and Directory Location

SurveySettings values are stored in a single file named Survey.settings in the settings folder. The .settings files are different from other named components because there is only one settings file for each settings component.

Version

SurveySettings is available in API version 47.0 and later.

Fields

Field Name	Field Type	Description
`enableSurvey`	boolean	Indicates whether Surveys is enabled for your org (true) or not (false). The default value is `false`.
`enableSurveyOwnerCanManageResponse`	boolean	Indicates whether the owner of a survey can manage its responses. The default value is `false`.

Declarative Metadata Sample Definition

The following is an example of a SurveySettings component.

```
<?xml version="1.0" encoding="UTF-8"?>
<SurveySettingsxmlns="http://soap.sforce.com/2006/04/metadata">
<enableSurvey>true</enableSurvey>
<enableSurveyOwnerCanManageResponse>false</enableSurveyOwnerCanManageResponse>
</SurveySettings>
```

The following is an example `package.xml` that references the previous definition.

```
<?xml version="1.0" encoding="UTF-8"?>
<Package xmlns="http://soap.sforce.com/2006/04/metadata">
    <types>
        <members>Survey</members>
        <name>Settings</name>
    </types>
    <version>47.0</version>
</Package>
```

Territory2Settings

Represents an org's Territory2 settings. Use Territory2 settings to set the access level that Territory Management 2.0 users have to records associated with sales territories, and to enable features. The standard record access settings apply to accounts and opportunities. If your Salesforce org uses *`Private`* default internal access for contacts or cases, you can also set access for those records. This type extends the Metadata metadata type and inherits its `fullName` field.

In the package manifest, all organization settings metadata types are accessed using the "Settings" name. See Settings for more details.

File Suffix and Directory Location

Territory2Settings values are stored in a single file named `Territory2.settings` in the `settings` directory of the corresponding package directory. The `.settings` files are different from other named components because there is only one settings file for each settings component.

Version

Territory2Settings is available in API version 32.0 and later.

Special Access Rules

The Territory2Model object has a `State` field in the SOAP API. States include `Planning`, `Active`, `Archived`, and a number of other states, such as `Cloning`, that indicate that a process is underway. Users who do not have the "Manage Territories" permission can access only territories that belong to the model in `Active` state. The "Manage Territories" permission is required for `deploy()` calls for all territory management entities, in addition to the "Modify All Data" permission required by Metadata API. Using `retrieve()` without the "Manage Territories" permission will return only entities that belong to a Territory2Model in `Active` state. We recommend against retrieving without the "Manage Territories" permission because the call will retrieve only partial data.

Fields

Field Name	Field Type	Description
`defaultAccountAccessLevel`	string	Sets the default level of access that users have to account records in territories: *`view`* and *`edit`* accounts assigned to territories or *`view`*, *`edit`*, *`transfer`*, and *`delete`* accounts assigned to territories.
`defaultCaseAccessLevel`	string	Sets the default level of access that users have to case records in territories: *`view`* and *`edit`* accounts assigned to territories or *`view`*, *`edit`*, *`transfer`*, and *`delete`* accounts assigned to territories.
`defaultContactAccessLevel`	string	Sets the default level of access that users have to contact records in territories: *`view`* and *`edit`* accounts assigned to territories or *`view`*, *`edit`*, *`transfer`*, and *`delete`* accounts assigned to territories.

Field Name	Field Type	Description
defaultOpportunityAccessLevel	string	Sets the default level of access that users have to opportunity records in territories: *view* and *edit* accounts assigned to territories or *view*, *edit*, *transfer*, and *delete* accounts assigned to territories.
enableTerritoryManagement2	boolean	Enables and disables Enterprise Territory Management only. If true, Enterprise Territory Management is enabled. If false (default), Enterprise Territory Management is not enabled. Enabling and disabling Enterprise Territory Management is exclusive of all other operations, and the field value must be true before other territory-management operations can run. Available in API version 47.0 and later.

Declarative Metadata Sample Definition

The following example shows the definition of a Territory2Settings component.

```
<?xml version="1.0" encoding="UTF-8"?>
<Territory2Settings xmlns="http://soap.sforce.com/2006/04/metadata">
    <defaultAccountAccessLevel>Owner</defaultAccountAccessLevel>
    <defaultOpportunityAccessLevel>Read</defaultOpportunityAccessLevel>
    <defaultCaseAccessLevel>None</defaultCaseAccessLevel>
    <defaultContactAccessLevel>Edit</defaultContactAccessLevel>
    <enableTerritoryManagement2>true</enableTerritoryManagement2>
</Territory2Settings>
```

Usage

Territory Management 2.0 components don't support packaging or change sets and aren't supported in CRUD calls.

Wildcard Support in the Manifest File

The wildcard character * (asterisk) in the package.xml manifest file doesn't apply to metadata types for feature settings. The wildcard applies only when retrieving all settings, not for an individual setting. For details, see Settings. For information about using the manifest file, see Deploying and Retrieving Metadata with the Zip File.

TrailheadSettings

Represents an org's access to myTrailhead. This type extends the Metadata metadata type and inherits its fullName field.

In the package manifest, all organization settings metadata types are accessed using the "Settings" name. See Settings for more details.

File Suffix and Directory Location

TrailheadSettings values are stored in the Trailhead.settings file in the settings directory. The .settings files are different from other named components because there is only one settings file for each settings component.

Version

TrailheadSettings components are available in API version 47.0 and later.

Special Access Rules

TrailheadSettings is available only in orgs that have myTrailhead enabled.

Fields

Field Name	Field Type	Description
`enableMyTrailheadPref`	boolean	Indicates whether a customer has access to myTrailhead (`true`) or not (`false`). The default value of this field is `true`.

Declarative Metadata Sample Definition

The following is an example of a TrailheadSettings component.

```
<?xml version="1.0" encoding="UTF-8"?>
<TrailheadSettings xmlns="http://soap.sforce.com/2006/04/metadata">
    <enableMyTrailheadPref>true</enableMyTrailheadPref>
</TrailheadSettings>
```

The following is an example `package.xml` that references the previous definition.

```
<?xml version="1.0" encoding="UTF-8"?>
<Package xmlns="http://soap.sforce.com/2006/04/metadata">
    <types>
        <members>Trailhead</members>
        <name>Settings</name>
    </types>
    <version>47.0</version>
</Package>
```

Wildcard Support in the Manifest File

The wildcard character `*` (asterisk) in the `package.xml` manifest file doesn't apply to metadata types for feature settings. The wildcard applies only when retrieving all settings, not for an individual setting. For details, see Settings. For information about using the manifest file, see Deploying and Retrieving Metadata with the Zip File.

TrialOrgSettings

Represents the settings in a trial user's org. This type extends the Metadata metadata type and inherits its `fullName` field.

In the package manifest, all organization settings metadata types are accessed using the "Settings" name. See Settings for more details.

File Suffix and Directory Location

TrialOrgSettings values are stored in the `TrialOrg.settings` file in the `settings` directory.The `.settings` files are different from other named components because there is only one settings file for each settings component.

Version

TrialOrgsettings is available in API version 48.0 and later.

Special Access Rules

Access to TrialOrgSettings requires users to complete the checkout flow in Enterprise, Professional, or Essentials editions. For Essentials, you can also access TrialOrgSettings by completing step 7 of the Setup Assistant.

Fields

Field Name	Field Type	Description
`enableSampleDataDeleted`	boolean	Indicates whether sample data may be deleted on trial orgs (`true`) or not (`false`). The default value is `false`.

Declarative Metadata Sample Definition

The following is an example of a TrialOrgSettings component.

```
<?xml version="1.0" encoding="UTF-8"?>
<TrialOrgSettings xmlns="http://soap.sforce.com/2006/04/metadata">
    <enableSampleDataDeleted>false</enableSampleDataDeleted>
</TrialOrgSettings>
```

Wildcard Support in the Manifest File

This metadata type doesn't support the wildcard character `*` (asterisk) in the `package.xml` manifest file. For information about using the manifest file, see Deploying and Retrieving Metadata with the Zip File.

UserEngagementSettings

Represents the metadata associated with various feature settings around Lightning Experience transition and adoption, user engagement and adoption assistance, and adoption apps. This type extends the Metadata metadata type and inherits its `fullName` field.

File Suffix and Directory Location

UserEngagementSettings components have the suffix `.settings` and are stored in the `settings` folder.

Version

Prompt components are available in API version 47.0 and later.

Special Access Rules

See related Salesforce Help for each feature for permission and edition requirements.

Fields

Field Name	Field Type	Description
canGovCloudUseAdoptionApps	boolean	Indicates whether orgs using the Government Orgs can access Lightning Experience Transition Tools external applications (`true`) or not (`false`). Examples of these applications are Salesforce Optimizer, Lightning Experience Transition Assistant, and the Lightning Experience Readiness Report. See **External Application Settings** under **Security** in Setup. The default is `false`.
doesScheduledSwitcherRunDaily	boolean	Indicates where users are automatically switched from Salesforce Classic to Lightning Experience every day (`true`) or weekly (`false`). If `false`, then users are switched weekly. The default is `false`. See Encourage Users to Stay in Lightning Experience in Salesforce Help.
enableCustomHelpGlobalSection	boolean	Indicates whether a custom section has been added to the Lightning Experience Help Menu (`true`) or not (`false`). The default is `false`. See Define Custom Help for the Lightning Experience Help Menu in Salesforce Help for more information.
enableHelpMenuShowFeedback	boolean	Indicates whether the Give Feedback to Salesforce link in the Lightning Experience Help Menu is visible to users (`true`) or not (`false`). The default is `true`. Even if `false`, admins always see all links in the Help Menu. See Define Custom Help for the Lightning Experience Help Menu in Salesforce Help for more information.
enableHelpMenuShowHelp	boolean	Indicates whether the Help For This Page section in the Lightning Experience Help Menu is visible to users (`true`) or not (`false`). The default is `true`. Even if `false`, admins always see all links in the Help Menu. See Define Custom Help for the Lightning Experience Help Menu in Salesforce Help for more information.
enableHelpMenuShowNewUser	boolean	Indicates whether the Getting Started section in the Lightning Experience Help Menu is visible to users (`true`) or not (`false`). The default is `true`. Even if `false`, admins always see all links in the Help Menu. See Define Custom Help for the Lightning Experience Help Menu in Salesforce Help for more information.
enableHelpMenuShowSearch	boolean	Indicates whether the Search Documentation link in the Lightning Experience Help Menu is visible to users (`true`) or not (`false`). The default is `true`. Even if `false`, admins always see all links in the Help Menu. See Define Custom Help for the Lightning Experience Help Menu in Salesforce Help for more information.
enableHelpMenuShowSfdcContent	boolean	Indicates whether any Salesforce-created help resources in Lightning Experience Help Menu are visible to users (`true`) or not (`false`). The default is `true`. Even if `false`, admins always see all links in the Help

Field Name	Field Type	Description
		Menu. See Define Custom Help for the Lightning Experience Help Menu in Salesforce Help for more information.
`enableHelpMenuShowShortcut`	boolean	Indicates whether the View Keyboard Shortcuts link in the Lightning Experience Help Menu is visible to users (`true`) or not (`false`). The default is `true`. Even if `false`, admins always see all links in the Help Menu. See Define Custom Help for the Lightning Experience Help Menu in Salesforce Help for more information.
`enableHelpMenuShowSupport`	boolean	Indicates whether the Get Support link in the Lightning Experience Help Menu is visible to users (`true`) or not (`false`). The default is `true`. Even if `false`, admins always see all links in the Help Menu. See Define Custom Help for the Lightning Experience Help Menu in Salesforce Help for more information.
`enableHelpMenuShowTrailhead`	boolean	Indicates whether the Go to Trailhead link in the Lightning Experience Help Menu is visible to users (`true`) or not (`false`). The default is `true`. Even if `false`, admins always see all links in the Help Menu. See Define Custom Help for the Lightning Experience Help Menu in Salesforce Help for more information.
`enableIBILOptOutDashboards`	boolean	Indicates whether the It's Better in Lightning prompt about Dashboards is hidden from users (`true`) or not (`false`). The default is `true`. See It's Better in Lightning Prompts in Salesforce Help for more information.
`enableIBILOptOutEvents`	boolean	Indicates whether the It's Better in Lightning prompt about Events/Calendar is hidden from users (`true`) or not (`false`). The default is `true`. See It's Better in Lightning Prompts in Salesforce Help for more information.
`enableIBILOptOutReports`	boolean	Indicates whether the It's Better in Lightning prompt about Reports is hidden from users (`true`) or not (`false`). The default is `true`. See It's Better in Lightning Prompts in Salesforce Help for more information.
`enableIBILOptOutTasks`	boolean	Indicates whether the It's Better in Lightning prompt about Tasks is hidden from users (`true`) or not (`false`). The default is `true`. See It's Better in Lightning Prompts in Salesforce Help for more information.
`enableLexToClassicFeedbackEnable`	boolean	Indicates whether the Switch to Salesforce Classic Feedback Form is shown to users (`true`) or not (`false`). The default is `false`. See Switch to Salesforce Classic Feedback Form in Salesforce Help for more information.
`enableOrchestrationInSandbox`	boolean	Indicates whether adoption assistance and other in-app guidance is shown to users in sandbox orgs (`true`) or not (`false`). The default is `false`. See Define Prompts in Lightning Experience in Salesforce Help for more information.
`enableOrgUserAssistEnabled`	boolean	Indicates whether all custom in-app guidance created by an org is shown to users (`true`) or not (`false`). Doesn't affect active status. The default

Field Name	Field Type	Description
		is `true`. See Define Prompts in Lightning Experience in Salesforce Help for more information.
`enableScheduledSwitcher`	boolean	Indicates whether users are automatically switched from Salesforce Classic to Lightning Experience (`true`) or not (`false`). The default is `true`. See Encourage Users to Stay in Lightning Experience in Salesforce Help.
`enableSfdcProductFeedbackSurvey`	boolean	Indicates whether the Salesforce Product Feedback Form is shown to users (`true`) or not (`false`). The default is `true`. See Salesforce Product Feedback Form in Salesforce Help for more information.
`enableShowSalesforceUserAssist`	boolean	Indicates whether all standard in-app guidance created by Salesforce is shown to users (`true`) or not (`false`). Doesn't affect active status. The default is `true`. See Define Prompts in Lightning Experience in Salesforce Help for more information.
`isCrucNotificationDisabled`	boolean	Indicates whether all notifications about the Winter '20 Turn on Lightning Experience critical update are hidden from admins (`true`) or not (`false`). The default is `false`.
`isLEXWelcomeMatDisabled`	boolean	Indicates whether the Lightning Experience welcome mat is hidden from users the first time they log into the user interface (`true`) or not (`false`). The default is `false`. See Lightning Experience Welcome Mat in Salesforce Help for more information.
`isMeetTheAssistantDisabledInClassic`	boolean	Indicates whether all notifications about the Lightning Experience Transition Assistant are hidden from admins in Salesforce Classic (`true`) or not (`false`). The default is `false`.
`isMeetTheAssistantDisabledInLightning`	boolean	Indicates whether all notifications about the Lightning Experience Transition Assistant are hidden from admins in Lightning Experience (`true`) or not (`false`). The default is `false`.
`optimizerAppEnabled`	boolean	Indicates whether Salesforce Optimizer is turned on in the org (`true`) or not (`false`). The default is `false`. See Improve Your Implementation with Salesforce Optimizer in Salesforce Help.

Declarative Metadata Sample Definition

The following is an example of a UserEngagementSettings component.

```
<?xml version="1.0" encoding="UTF-8"?>
<UserEngagementSettings xmlns="http://soap.sforce.com/2006/04/metadata">
   <canGovCloudUseAdoptionApps>false</canGovCloudUseAdoptionApps>
   <doesScheduledSwitcherRunDaily>true</doesScheduledSwitcherRunDaily>
   <enableCustomHelpGlobalSection>true</enableCustomHelpGlobalSection>
   <enableHelpMenuShowFeedback>true</enableHelpMenuShowFeedback>
   <enableHelpMenuShowHelp>true</enableHelpMenuShowHelp>
   <enableHelpMenuShowNewUser>true</enableHelpMenuShowNewUser>
   <enableHelpMenuShowSearch>true</enableHelpMenuShowSearch>
   <enableHelpMenuShowSfdcContent>true</enableHelpMenuShowSfdcContent>
   <enableHelpMenuShowShortcut>true</enableHelpMenuShowShortcut>
   <enableHelpMenuShowSupport>true</enableHelpMenuShowSupport>
   <enableHelpMenuShowTrailhead>true</enableHelpMenuShowTrailhead>
   <enableIBILOptOutDashboards>true</enableIBILOptOutDashboards>
   <enableIBILOptOutEvents>true</enableIBILOptOutEvents>
   <enableIBILOptOutReports>true</enableIBILOptOutReports>
   <enableIBILOptOutTasks>true</enableIBILOptOutTasks>
   <enableLexToClassicFeedbackEnable>true</enableLexToClassicFeedbackEnable>
   <enableOrgUserAssistEnabled>true</enableOrgUserAssistEnabled>
   <enableScheduledSwitcher>true</enableScheduledSwitcher>
   <enableSfdcProductFeedbackSurvey>true</enableSfdcProductFeedbackSurvey>
   <enableOrchestrationInSandbox>true</enableOrchestrationInSandbox>
   <enableShowSalesforceUserAssist>true</enableShowSalesforceUserAssist>
   <isCrucNotificationDisabled>false</isCrucNotificationDisabled>
   <isLEXWelcomeMatDisabled>false</isLEXWelcomeMatDisabled>
   <isMeetTheAssistantDisabledInClassic>false</isMeetTheAssistantDisabledInClassic>
   <isMeetTheAssistantDisabledInLightning>false</isMeetTheAssistantDisabledInLightning>
   <optimizerAppEnabled>true</optimizerAppEnabled>
</UserEngagementSettings>
```

The following is an example `package.xml` that references the previous definition.

```
<?xml version="1.0" encoding="UTF-8"?>
<Package xmlns="http://soap.sforce.com/2006/04/metadata">
   <types>
      <members>UserEngagement</members>
      <name>Settings</name>
   </types>
   <version>47.0</version>
</Package>
```

Wildcard Support in the Manifest File

The wildcard character `*` (asterisk) in the `package.xml` manifest file doesn't apply to metadata types for feature settings. The wildcard applies only when retrieving all settings, not for an individual setting. For details, see Settings. For information about using the manifest file, see Deploying and Retrieving Metadata with the Zip File.

UserInterfaceSettings

Represents the settings that modify the behavior of the org's user interface. This type extends the Metadata metadata type and inherits its `fullName` field.

In the package manifest, all organization settings metadata types are accessed using the "Settings" name. See Settings for more details.

File Suffix and Directory Location

A UserInterfaceSettings component file has the suffix `.settings` and is stored in the `settings` directory. The `.settings` files are different from other named components because there is only one settings file for each settings component.

Version

UserInterfaceSettings components are available in API version 46.0 and later.

Fields

Field Name	Field Type	Description
`enableAsyncRelatedLists`	boolean	Indicates whether related lists are loaded asynchronously (`true`) or not (`false`). The default is `false`. Available in API version 47.0 and later. Salesforce Classic only.
`enableClickjackUserPageHeaderless`	boolean	Indicates whether a Visualforce page that hides the standard header has clickjack protections (`true`) or not (`false`). The default is `true`. This setting applies to all of your Visualforce pages.
`enableCollapsibleSections`	boolean	Indicates whether users are allowed to collapse or expand sections in record details by using the arrow icon next to the section heading. The default is `true`.
`enableCollapsibleSidebar`	boolean	Indicates whether users are allowed to show or hide the sidebar on every page that normally includes it (`true`) or not (`false`). The default is `false`. Applies to Salesforce Classic only.
`enableCustomObjectTruncate`	boolean	Indicates whether users with Customize Application permission can truncate custom objects (`true`) or not (`false`). When you truncate an object, you delete the object's associated records permanently, while preserving the empty object and its metadata. The default is `false`. Available in API version 47.0 and later.
`enableCustomSidebarOnAllPages`	boolean	Indicates whether custom sidebar components are available on all pages for all org users (`true`) or not (`false`). The default is `false`. Applies to Salesforce Classic only.
`enableDeleteFieldHistory`	boolean	Indicates whether users can delete field history and field history archive records (`true`) or not (`false`). The default is `false`. Available in API version 47.0 and later.
`enableExternalObjectAsyncRelatedLists`	boolean	Indicates whether related lists of external objects are loaded asynchronously (`true`) or not (`false`). The default is `true`. Available in API version 48.0 and later. Salesforce Classic only.
`enableHoverDetails`	boolean	Indicates whether an interactive overlay containing record details is displayed (`true`) or not (`false`). The default is `true`. Note: To view hover details for a record, users need the appropriate sharing access and field-level security access for the fields in the mini page layout.
`enableInlineEdit`	boolean	Indicates whether users are allowed to edit field values on a record's detail page (`true`) or not (`false`). The default is `true`.

Field Name	Field Type	Description
`enablePersonalCanvas`	boolean	Indicates whether users can install and use personal canvas apps `(true)` or not `(false)`. The default is `true`. This setting applies to all of your Visualforce pages.
`enableQuickCreate`	boolean	Indicates whether an area displays on a tab home page (corresponds to the **Show Quick Create** setting), allowing users to create a record quickly with minimal information (`true`) or not (`false`). The Quick Create area displays by default on the tab home pages for leads, accounts, contacts, and opportunities. You can control whether the Quick Create area is displayed on all relevant tab home pages.

Declarative Metadata Sample Definition

The following is an example of a UserInterfaceSettings component.

```
<?xml version="1.0" encoding="UTF-8"?>
<UserInterfaceSettings xmlns="http://soap.sforce.com/2006/04/metadata">
  <enableDeleteFieldHistory>false</enableDeleteFieldHistory>
  <enableInlineEdit>true</enableInlineEdit>
  <enableHoverDetails>false</enableHoverDetails>
  <enableQuickCreate>true</enableQuickCreate>
  <enablePersonalCanvas>false</enablePersonalCanvas>
  <enableClickjackUserPageHeaderless>true</enableClickjackUserPageHeaderless>
</UserInterfaceSettings>
```

Example Package Manifest

The following is an example package manifest used to deploy or retrieve the user interface settings metadata for an organization:

```
<?xml version="1.0" encoding="UTF-8"?>
<Package xmlns="http://soap.sforce.com/2006/04/metadata">
 <types>
  <members>UserInteface</members>
  <name>Settings</name>
 </types>
 <version>46.0</version>
</Package>
```

Wildcard Support in the Manifest File

The wildcard character `*` (asterisk) in the `package.xml` manifest file doesn't apply to metadata types for feature settings. The wildcard applies only when retrieving all settings, not for an individual setting. For details, see Settings. For information about using the manifest file, see Deploying and Retrieving Metadata with the Zip File.

UserManagementSettings

Represents a selection of user management options that appear on the User Management Settings Setup page. This type extends the Metadata metadata type and inherits its `fullName` field.

In the package manifest, all organization settings metadata types are accessed using the "Settings" name. See Settings for more details.

File Suffix and Directory Location

UserMangementSettings are stored in the `UserManagement.settings` directory. The `.settings` files are different from other named components because there is only one settings file for each settings component.

Version

Manage org-wide settings for certain options. User Management Settings are available in API version 46.0 and later.

Fields

Field	Field Type	Description
`enableContactlessExternalIdentityUsers`	boolean	If you enable **Enable Contactless External Identity Users** (`true`), and your org has the External Identity User license, you can create contactless users. Having external identity users without contact information reduces the overhead of managing external identity users. The External Identity User license is available to customers who purchase Identity for Customers and Partners. The default is not enabled (`false`). Available in API version 47.0 and later.
`enableEnhancedPermsetMgmt`	boolean	If you enable **Enhanced Permission Set Component Views** (`true`), you can work with permission sets more easily. For example, when you have large numbers of Apex class assignments for permission sets, you can enable a paginated result set, standard filtering, and sorting.
`enableEnhancedProfileMgmt`	boolean	If you enable **Enhanced Profile Lists Views** (`true`), you can quickly view, customize, and edit list data .
`enableNewProfileUI`	boolean	If you enable **Enhanced Profile User Interface** (`true`), you can use the streamlined, enhanced profile user interface to browse, search, and modify settings. You can use only one user interface at a time.
`enableUserSelfDeactivate`	boolean	If you enable **User Self Deactivate** (`true`), users can deactivate their Community or Chatter accounts.
`enableScrambleUserData`	boolean	If you enable **Let Users Scramble Their User Data** (`true`), users can request that Salesforce remove all their personal data. Because Salesforce can't delete information, it scrambles their data. Scrambling a user's data is unrecoverable. So this org-wide setting serves as an extra precaution. If a user requests it, you scramble the data programmatically with the `obfuscateUser`

Field	Field Type	Description
		Apex method. You can use the method, for example, in a custom Apex trigger, workflow, or the Developer Console. This field is available in API version 47.0 and later.

Declarative Metadata Sample Definition

The following is an example of a UserManagementSettings component.

```
<?xml version="1.0" encoding="UTF-8"?>
<UserManagementSettings xmlns="http://soap.sforce.com/2006/04/metadata"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
    <enableCanAnswerContainUsername>true</enableCanAnswerContainUsername>
    <enableCanSaveUserPerm xsi:nil="true"/>
    <enableConcealPersonalInfo>true</enableConcealPersonalInfo>
    <enableContactlessExternalIdentityUsers>false</enableContactlessExternalIdentityUsers>

    <enableEnhancedPermsetMgmt>false</enableEnhancedPermsetMgmt>
    <enableEnhancedProfileMgmt>true</enableEnhancedProfileMgmt>
    <enableNewProfileUI>false</enableNewProfileUI>
    <enableScrambleUserData>false</enableScrambleUserData>
    <enableUserSelfDeactivate>false</enableUserSelfDeactivate>
</UserManagementSettings>
```

The following is an example `package.xml` manifest that references the previous definition.

```
<?xml version="1.0" encoding="UTF-8"?>
<Package xmlns="http://soap.sforce.com/2006/04/metadata">
    <types>
        <members>UserManagement</members>
        <name>Settings</name>
    </types>
    <version>47.0</version>
</Package>
```

Wildcard Support in the Manifest File

The wildcard character `*` (asterisk) in the `package.xml` manifest file doesn't apply to metadata types for feature settings. The wildcard applies only when retrieving all settings, not for an individual setting. For details, see Settings. For information about using the manifest file, see Deploying and Retrieving Metadata with the Zip File.

VoiceSettings

Represents an org's Lightning Dialer settings, such as call recording, conferencing, and voicemail. This type extends the Metadata metadata type and inherits its `fullName` field.

In the package manifest, all organization settings metadata types are accessed using the "Settings" name. See Settings for more details.

File Suffix and Directory Location

VoiceSettings values are stored in the `Voice.settings` file in the `settings` directory. The `.settings` files are different from other named components because there is only one settings file for each settings component.

Version

VoiceSettings is available in API version 47.0 and later.

Fields

Field Name	Field Type	Description
`enableCallDisposition`	boolean	Indicates whether call disposition is enabled (`true`) or not (`false`). With call disposition, also called Call Result, sales reps can track whether a call was connected and how it went. Default value is `false`. To use this feature, enable Dialer in Lightning Experience.
`enableVoiceCallList`	boolean	Indicates whether Call List is enabled (`true`) or not (`false`). Sales reps can use call list to keep a running list of the calls they want to make. Default value is `false`. To use this feature, enable Dialer in Lightning Experience.
`enableVoiceCallRecording`	boolean	Indicates whether Call Recording is enabled (`true`) or not (`false`). Sales reps can record important calls directly from the call panel in Lightning Dialer. Default value is `false`. To use this feature, enable Dialer in Lightning Experience.
`enableVoiceCoaching`	boolean	Indicates whether Call Monitoring is enabled (`true`) or not (`false`). Using the Monitor tab in the call panel, managers can listen to the calls of their sales reps for personalized coaching. Default value is `false`. To use this feature, enable Dialer in Lightning Experience.
`enableVoiceConferencing`	boolean	Reserved for future use.
`enableVoiceLocalPresence`	boolean	Indicates whether Local Presence is enabled (`true`) or not (`false`). Local Presence displays phone numbers with the same area code as the prospects your reps are calling, so more calls are answered. Default value is `false`. To use this feature, enable Dialer in Lightning Experience.

Field Name	Field Type	Description
`enableVoiceMail`	boolean	Indicates whether voicemail is enabled (`true`) or not (`false`). Sales reps can receive and store up to 20 personal voicemail messages in Salesforce. Default value is `false`. To use this feature, enable Dialer in Lightning Experience.
`enableVoiceMailDrop`	boolean	Indicates whether Voicemail Drop is enabled (`true`) or not (`false`). Sales reps can "drop" (or send) prerecorded messages to recipients' voicemail boxes. Default value is `false`. To use this feature, enable Dialer in Lightning Experience.

Declarative Metadata Sample Definition

The following is an example of the package file.

```
<?xml version="1.0" encoding="UTF-8"?>
<Package xmlns="http://soap.sforce.com/2006/04/metadata">
    <types>
    </types>
    <version>28.0</version>
</Package>
```

The package file references the following Voice.settings file.

```
<?xml version="1.0" encoding="UTF-8"?>
<VoiceSettings xmlns="http://soap.sforce.com/2006/04/metadata">
    <enableCallDisposition>true</enableCallDisposition>
    <enableVoiceCallList>true</enableVoiceCallList>
    <enableVoiceCallRecording>true</enableVoiceCallRecording>
    <enableVoiceCoaching>true</enableVoiceCoaching>
    <enableVoiceConferencing>true</enableVoiceConferencing>
    <enableVoiceLocalPresence>true</enableVoiceLocalPresence>
    <enableVoiceMail>true</enableVoiceMail>
    <enableVoiceMailDrop>true</enableVoiceMailDrop>
</VoiceSettings>
```

Wildcard Support in the Manifest File

The wildcard character `*` (asterisk) in the `package.xml` manifest file doesn't apply to metadata types for feature settings. The wildcard applies only when retrieving all settings, not for an individual setting. For details, see Settings. For information about using the manifest file, see Deploying and Retrieving Metadata with the Zip File.

WorkDotComSettings

Represents WorkDotCom settings. This type extends the Metadata metadata type and inherits its `fullName` field.

File Suffix and Directory Location

WorkDotComSettings components are stored in the files `udd-work.xml`, `udd-work-certifications.xml`, `udd-work-upgrade.xml` and `udd-work-feedback.xml`. You can find these files in the folder `udd-xml/resource/udd`.

Version

WorkDotComSettings components are available in API version 31.0 and later.

Fields

Field Name	Field Type	Description
`enableCoachingManagerGroupAccess`	boolean	Indicates whether Coaching Manager Group Access is available to users `(true)` or not `(false)`. Default value is `true`. Deprecated.
`enableGoalManagerGroupAccess`	boolean	Indicates whether Goal Manager Group Access is available to users `(true)` or not `(false)`. Default value is `true`. Deprecated.
`enableProfileSkills`	boolean	Indicates whether Profile Skills is available to users `(true)` or not `(false)`. Default value is `true`.
`enableProfileSkillsAddFeedPost`	boolean	Indicates whether Add Skills as Feed Posts is available to users `(true)` or not `(false)`. Default value is `true`.
`enableProfileSkillsAutoSuggest`	boolean	Indicates whether Profile Skills Auto Suggest is available to users `(true)` or not `(false)`. Default value is `true`.
`enableProfileSkillsUsePlatform`	boolean	Indicates whether Profile Skills Use Platform is available to users `(true)` or not `(false)`. Default value is `true`.
`enableWorkBadgeDefRestrictPref`	boolean	Indicates whether Badge Definition Restriction is available to users `(true)` or not `(false)`. Default value is `true`. Deprecated.
`enableWorkCalibration`	boolean	Indicates whether Calibration is available to users `(true)` or not `(false)`. Default value is `false`. Deprecated.
`enableWorkCanvasPref`	boolean	Indicates whether Canvas is available to users `(true)` or not `(false)`. Default value is `true`. Deprecated.
`enableWorkCertification`	boolean	Indicates whether Certification is available to users `(true)` or not `(false)`. Default value is `true`. Deprecated.

Field Name	Field Type	Description
enableWorkCertificationNotification	boolean	Indicates whether Certification Notification is available to users (true) or not (false). Default value is false. Deprecated.
enableWorkRewardsPref	boolean	Indicates whether Rewards is available to users (true) or not (false). Default value is true. Deprecated.
enableWorkThanksPref	boolean	Indicates whether Thanks is available to users (true) or not (false). Default value is true.
enableWorkUseObjectivesForGoals	boolean	Indicates whether the app willl use Objectives as Synonyms for Goals (true) or not (false). Default value is false. Deprecated.

Declarative Metadata Sample Definition

The following is an example of a WorkDotComSettings component.

```
<?xml version="1.0" encoding="UTF-8"?>
<WorkDotComSettings xmlns="http://soap.sforce.com/2006/04/metadata">
    <enableCoachingManagerGroupAccess>true</enableCoachingManagerGroupAccess>
    <enableGoalManagerGroupAccess>true</enableGoalManagerGroupAccess>
    <enableProfileSkills>true</enableProfileSkills>
    <enableProfileSkillsAddFeedPost>true</enableProfileSkillsAddFeedPost>
    <enableProfileSkillsAutoSuggest>true</enableProfileSkillsAutoSuggest>
    <enableProfileSkillsUsePlatform>true</enableProfileSkillsUsePlatform>
    <enableWorkBadgeDefRestrictPref>true</enableWorkBadgeDefRestrictPref>
    <enableWorkCalibration>true</enableWorkCalibration>
    <enableWorkCanvasPref>true</enableWorkCanvasPref>
    <enableWorkCertification>true</enableWorkCertification>
    <enableWorkCertificationNotification>true</enableWorkCertificationNotification>
    <enableWorkRewardsPref>true</enableWorkRewardsPref>
    <enableWorkThanksPref>true</enableWorkThanksPref>
    <enableWorkUseObjectivesForGoals>true</enableWorkUseObjectivesForGoals>
</WorkDotComSettings>
```

SharedTo

SharedTo defines the sharing access for a list view or a folder. It can be used to specify the target and source for owner-based sharing rules.

See “Sharing Considerations” and “What Is a Group?” in Salesforce Help.

Note: SharedTo is included in the metadata for shared and private list views. SharedTo isn’t in the metadata for public list views.

Declarative Metadata File Suffix and Directory Location

SharedTo is used with ListView, Folder, and SharingRules.

Version

SharedTo is available in API version 17.0 and later.

Fields

Field	Field Type	Description
`allCustomerPortalUsers`	string	A group containing all customer portal users. This field is available in API version 24.0 and later.
`allInternalUsers`	string	A group containing all internal and nonportal users. This field is available in API version 24.0 and later.
`allPartnerUsers`	string	A group containing all partner users. This field is available in API version 24.0 and later.
`channelProgramGroup`	string	A system-managed group with sharing access containing all partner members of the corresponding channel program or level. This field is available in API version 41.0 and later.
`channelProgramGroups`	string[]	A list of system-managed groups with sharing access containing all partner members of the corresponding channel programs or levels. This field is available in API version 41.0 and later.
`group`	string[]	A list of groups with sharing access. Use this field instead of the `groups` field. This field is available in API version 22.0 and later.
`guestUser`	string[]	A list of guest user nicknames with sharing access. This field can be used only with SharingGuestRule. This field is available in API version 47.0 and later.
`groups`	string[]	A list of groups with sharing access. Use the `group` field instead for API version 22.0 and later.
`managerSubordinates`	string[]	A list of users whose direct and indirect subordinates receive sharing access. This field is available in API version 24.0 and later.
`managers`	string[]	A list of users whose direct and indirect managers receive sharing access. This field is available in API version 24.0 and later.

Field	Field Type	Description
portalRole	string[]	A list of groups with sharing access containing all users in a portal role. This field is available in API version 24.0 and later.
portalRoleandSubordinates	string[]	A list of groups with sharing access containing all users in a portal role or those under that role. This field is available in API version 24.0 and later.
role	string[]	A list of roles with sharing access. Use this field instead of the roles field. This field is available in API version 22.0 and later.
roleAndSubordinates	string[]	A list of roles with sharing access. All roles below each of these roles in the role hierarchy also have sharing access. If portal accounts are enabled, then all roles and portal accounts below each of these roles in the role hierarchy also have sharing access. Use this field instead of the rolesAndSubordinates field. This field is available in API version 22.0 and later.
roleAndSubordinatesInternal	string[]	A list of roles with sharing access. All roles below each of these roles in the role hierarchy also have sharing access. This field is available in API version 22.0 and later.
roles	string[]	A list of roles with sharing access. Use the role field instead for API version 22.0 and later.
rolesAndSubordinates	string[]	A list of roles with sharing access. All roles below each of these roles in the role hierarchy also have sharing access. If portal accounts are enabled, then all roles and portal accounts below each of these roles in the role hierarchy also have sharing access. Use the roleAndSubordinates field instead for API version 22.0 and later.
territories	string[]	A list of territories with sharing access. Use the territory field instead for API version 22.0 and later.
territoriesAndSubordinates	string[]	A list of territories with sharing access. All territories below each of these territories in the territory hierarchy also have sharing access. Use the territoryAndSubordinates field instead for API version 22.0 and later.

Field	Field Type	Description
`territory`	string[]	A list of territories with sharing access. Use this field instead of the `territories` field. If you're using Enterprise Territory Management, use *`modelName.territoryName`* for the shared-to and shared-from `territory` values, where: • *`modelName`* equals the name of the active territory model in the API. • *`territoryName`* equals the territory's name in the API. This field is available in API version 22.0 and later.
`territoryAndSubordinates`	string[]	A list of territories with sharing access. All territories below each of these territories in the territory hierarchy also have sharing access. Use this field instead of the `territoriesAndSubordinates` field. If you're using Enterprise Territory Management, use *`modelName.territoryName`* for the shared-to and shared-from `territoryAndSubordinates` values, where: • *`modelName`* equals the name of the active territory model in the API. • *`territoryName`* equals the territory's name in the API. This field is available in API version 22.0 and later.
`queue`	string[]	A list of queues with sharing access. Applies only to lead, case, and CustomObject sharing rules. This field is available in API version 24.0 and later.

SharingBaseRule

Represents sharing rule settings such as access level and to whom access is granted.

This type extends the Metadata metadata type and inherits its `fullName` field.

Note: You can't create a SharingBaseRule component directly. Use the components under SharingRules instead.

Version

SharingBaseRule replaces BaseSharingRule and is available in API version 33.0 and later.

Fields

Field	Field Type	Description
`accessLevel`	string	Required. The access level that the sharing rule grants.
`accountSettings`	AccountSharingRuleSettings[]	The access level for the account's children (case, contact, and opportunity).
`description`	string	Describes the sharing rule. Maximum of 1000 characters.
`label`	string	Required. Label for the sharing rule.
`sharedTo`	SharedTo	Required. Specifies who the record should be shared with.

AccountSharingRuleSettings

Defines the access level for the case, contact, and opportunity associated with the account.

Field	Field Type	Description
`caseAccessLevel`	string	Required. The access level that the user or group has to cases associated with the account. Possible values are: • None • Read • Edit
`contactAccessLevel`	string	Required. The access level that the user or group has to contacts associated with the account. Possible values are: • None • Read • Edit
`opportunityAccessLevel`	string	Required. The access level that the user or group has to opportunities associated with the account. Possible values are: • None • Read • Edit

Wildcard Support in the Manifest File

This metadata type doesn't support the wildcard character `*` (asterisk) in the `package.xml` manifest file. For information about using the manifest file, see Deploying and Retrieving Metadata with the Zip File.

SharingRules

Represents the base container for sharing rules, which can be criteria-based, ownership-based, territory-based, or for guest user access. SharingRules enables you to share records with a set of users, using rules that specify the access level for the target user group.

This type extends the Metadata metadata type and inherits its `fullName` field. For more information, see "Sharing Rules" in the Salesforce online help.

In API version 33.0 and later, retrieving, deleting, or deploying of all sharing rules in an organization is available. Wildcard support is also available. You can't retrieve, delete, or deploy manual sharing rules or sharing rules by their type (owner, criteria-based, territory, or guest user).

Declarative Metadata File Suffix and Directory Location

In API version 33.0 and later, components are stored in the sharingRules folder and their file name matches the object name with the suffix `.sharingRules`. Criteria-based, owner-based, territory-based, and guest user sharing rules are all contained in a `object.sharingRule` file.

Prior to API version 33.0, SharingRules components are stored in their corresponding object directory and the file name matches the object name. For example, the `accountSharingRules` directory contains an `Account.sharingRules` file for account sharing rules. SharingRules for custom objects are stored in the `customObjectSharingRules` directory, which contains files with the `.sharingRules` extension such as `ObjA__c.sharingRules`, where ObjA refers to the developer name of a custom object type.

Version

SharingRules components are available in API version 24.0 and later, but these components are no longer available in API version 33.0 and later: AccountSharingRules, CampaignSharingRules, CaseSharingRules, ContactSharingRules, LeadSharingRules, OpportunitySharingRules, AccountTerritorySharingRules, CustomObjectSharingRules, UserSharingRules.

In API version 33.0 and later, use SharingCriteriaRule, SharingOwnerRule and SharingTerritoryRule.

Special Access Rules

As of Spring '20 and later, only users with the View Setup and Configuration permission can access this object, and only users with the Manage Sharing permission can edit this object.

Fields

The following information assumes that you are familiar with implementing sharing rules for standard objects and custom objects. For more information on these fields, see "Sharing Settings" in the Salesforce online help.

Field	Field Type	Description
`sharingCriteriaRules`	SharingCriteriaRule[]	An array of criteria-based sharing rules. Available in API version 33.0 and later.
`sharingGuestRules`	SharingGuestRule[]	An array of guest user sharing rules. Available in API version 47.0 and later.
`sharingOwnerRules`	SharingOwnerRule[]	An array of ownership-based sharing rules. Available in API version 33.0 and later.
`sharingTerritoryRules`	SharingTerritoryRule[]	An array of territory-based sharing rules. Available in API version 33.0 and later.

SharingCriteriaRule

Defines a criteria-based sharing rule. It extends SharingBaseRule and inherits all its fields. Available in API version 33.0 and later.

Field	Field Type	Description
`booleanFilter`	string	Advanced filter conditions that are specified for the sharing rule.
`criteriaItems`	FilterItem[]	An array of the boolean criteria (conditions) for the sharing rule.

SharingGuestRule

Defines a guest user sharing rule. It extends SharingBaseRule and inherits all its fields, except `accountSettings`. Available in API version 47.0 and later.

Note: For SharingGuestRule, the `accessLevel` field can be set only to `Read`.

Field	Field Type	Description
`booleanFilter`	string	Advanced filter conditions that are specified for the sharing rule. Available in API version 48.0 and later.
`criteriaItems`	FilterItem[]	An array of the boolean criteria (conditions) for the sharing rule. Available in API version 48.0 and later.

SharingOwnerRule

Defines a ownership-based sharing rule. It extends SharingBaseRule and inherits all its fields. Available in API version 33.0 and later.

Field	Field Type	Description
`sharedFrom`	SharedTo	Required. Specifies the record owners. If you're using Enterprise Territory Management, use *`modelName.territoryName`* for the shared-to and shared-from `territory` and `territoryAndSubordinates` values on the SharedTo type, where: • *`modelName`* equals the name of the active territory model in the API. • *`territoryName`* equals the territory's name in the API.

SharingTerritoryRule

Defines a territory-based sharing rule. It extends SharingOwnerRule and inherits all its fields. Available in API version 33.0 and later.

AccountSharingRules

Represents the sharing rules for accounts. It extends the SharingRules metadata type and inherits its `fullName` field. Only available in API version 32.0 and earlier.

Field	Field Type	Description
`criteriaBasedRules`	AccountCriteriaBasedSharingRule[]	List that defines user criteria-based rules.
`ownerRules`	AccountOwnerSharingRule[]	List that defines user membership-based rules.

CampaignSharingRules

Represents the sharing rules for campaigns. It extends the SharingRules metadata type and inherits its `fullName` field. Only available in API version 32.0 and earlier.

Field	Field Type	Description
`criteriaBasedRules`	CampaignCriteriaBasedSharingRule[]	List that defines user criteria-based rules.
`ownerRules`	CampaignOwnerSharingRule[]	List that defines user membership-based rules.

CaseSharingRules

Represents the sharing rules for cases. It extends the SharingRules metadata type and inherits its `fullName` field. Only available in API version 32.0 and earlier.

Field	Field Type	Description
`criteriaBasedRules`	CaseCriteriaBasedSharingRule[]	List that defines user criteria-based rules.

Field	Field Type	Description
`ownerRules`	CaseOwnerSharingRule[]	List that defines user membership-based rules.

ContactSharingRules

Represents the sharing rules for contacts. It extends the SharingRules metadata type and inherits its `fullName` field. Only available in API version 32.0 and earlier.

Field	Field Type	Description
`criteriaBasedRules`	ContactCriteriaBasedSharingRule[]	List that defines user criteria-based rules.
`ownerRules`	ContactOwnerSharingRule[]	List that defines user membership-based rules.

LeadSharingRules

Represents the sharing rules for leads. It extends the SharingRules metadata type and inherits its `fullName` field. Only available in API version 32.0 and earlier.

Field	Field Type	Description
`criteriaBasedRules`	LeadCriteriaBasedSharingRule[]	List that defines user criteria-based rules.
`ownerRules`	LeadOwnerSharingRule[]	List that defines user membership-based rules.

OpportunitySharingRules

Represents the sharing rules for opportunities. It extends the SharingRules metadata type and inherits its `fullName` field. Only available in API version 32.0 and earlier.

Field	Field Type	Description
`criteriaBasedRules`	OpportunityCriteriaBasedSharingRule[]	List that defines user criteria-based rules.
`ownerRules`	OpportunityOwnerSharingRule[]	List that defines user membership-based rules.

AccountTerritorySharingRules

Represents the sharing rules for account territories in the original territory management feature. It extends the SharingRules metadata type and inherits its `fullName` field. Only available in API version 32.0 and earlier.

Field	Field Type	Description
`rules`	AccountTerritorySharingRule[]	List that defines user membership-based rules. The list of acceptable values for the `sharedFrom` fields are: • `territory` • `territoryAndSubordinates`

CustomObjectSharingRules

Represents the sharing rules for custom objects. It extends the SharingRules metadata type and inherits its `fullName` field. Only available in API version 32.0 and earlier.

Field	Field Type	Description
`criteriaBasedRules`	CustomObjectCriteriaBasedSharingRule[]	List that defines user criteria-based rules.
`ownerRules`	CustomObjectOwnerSharingRule[]	List that defines user membership-based rules.

UserSharingRules

Represents the sharing rules for users. With user sharing rules, you can share members of a group with members of another group. It extends the SharingRules metadata type and inherits its `fullName` field. Only available in API version 32.0 and earlier.

Field	Field Type	Description
`criteriaBasedRules`	UserCriteriaBasedSharingRule[]	List that defines user criteria-based rules.
`membershipRules`	UserMembershipSharingRule[]	List that defines user membership-based rules.

Declarative Metadata Sample Definition

For retrieving sharing rules, see `package.xml` sample at Sharing Rules.

The following sample XML definition represents a criteria-based sharing rule in API version 33.0.

```
<?xml version="1.0" encoding="UTF-8"?>
<SharingRules xmlns="http://soap.sforce.com/2006/04/metadata">
    <sharingCriteriaRules>
        <fullName>AccountCriteriaShareWithCEO</fullName>
        <accessLevel>Edit</accessLevel>
        <accountSettings>
            <caseAccessLevel>Read</caseAccessLevel>
            <contactAccessLevel>Edit</contactAccessLevel>
            <opportunityAccessLevel>Edit</opportunityAccessLevel>
        </accountSettings>
        <criteriaItems>
            <field>Name</field>
            <operation>startsWith</operation>
            <value>Test</value>
        </criteriaItems>
        <description>my account criteria rule description</description>
        <label>AccountCriteriaShareWithCEO</label>
        <sharedTo>
            <role>CEO</role>
        </sharedTo>
    </sharingCriteriaRules>
</SharingRules>
```

The following sample XML definition represents an ownership-based sharing rule in API version 33.0.

```
<?xml version="1.0" encoding="UTF-8"?>
<SharingRules xmlns="http://soap.sforce.com/2006/04/metadata">
    <sharingOwnerRules>
        <fullName>MyCase</fullName>
        <accessLevel>Edit</accessLevel>
        <description>my case test owner sharing rule desc</description>
        <label>MyCase</label>
        <sharedFrom>
            <role>COO</role>
        </sharedFrom>
        <sharedTo>
            <role>CEO</role>
        </sharedTo>
    </sharingOwnerRules>
</SharingRules>
```

The following sample XML definition represents a territory-based sharing rule in API version 33.0.

```
<?xml version="1.0" encoding="UTF-8"?>
<SharingRules xmlns="http://soap.sforce.com/2006/04/metadata">
    <sharingTerritoryRules>
        <fullName>MyAccountTerritoryRule</fullName>
        <accessLevel>Read</accessLevel>
        <accountSettings>
            <caseAccessLevel>None</caseAccessLevel>
            <contactAccessLevel>Read</contactAccessLevel>
            <opportunityAccessLevel>None</opportunityAccessLevel>
        </accountSettings>
        <description>MyAccountTerritoryRule desc</description>
        <label>MyAccountTerritoryRule</label>
        <sharedFrom>
            <territory>My_territory</territory>
        </sharedFrom>
        <sharedTo>
            <role>CEO</role>
        </sharedTo>
    </sharingTerritoryRules>
</SharingRules>
```

The following is the definition of two account owner-based sharing rules in API version 32.0 and earlier. The file name corresponds to `Account.sharingRules` under the `accountSharingRules` directory. In this definition, ownerRules corresponds to AccountOwnerSharingRule.

```
<?xml version="1.0" encoding="UTF-8"?>
<AccountSharingRules xmlns="http://soap.sforce.com/2006/04/metadata">
    <ownerRules>
        <fullName>G1Dev_G2New</fullName>
        <sharedFrom>
            <group>G1Dev</group>
        </sharedFrom>
        <sharedTo>
            <group>G2New</group>
        </sharedTo>
        <accountAccessLevel>Read</caseAccessLevel>
        <caseAccessLevel>None</caseAccessLevel>
        <contactAccessLevel>Read</contactAccessLevel>
        <name>G1Dev_G2New</name>
        <opportunityAccessLevel>Edit</opportunityAccessLevel>
    </ownerRules>
    <ownerRules>
        <fullName>G2New_R1New</fullName>
        <sharedFrom>
            <group>G2New</group>
        </sharedFrom>
        <sharedTo>
            <roleAndSubordinates>R1New</roleAndSubordinates>
        </sharedTo>
        <accountAccessLevel>Edit</accountAccessLevel>
        <caseAccessLevel>Read</caseAccessLevel>
        <contactAccessLevel>Edit</contactAccessLevel>
        <name>G2New_R1New</name>
        <opportunityAccessLevel>None</opportunityAccessLevel>
    </ownerRules>
</AccountSharingRules>
```

The following is the definition of a user criteria-based sharing rule and a user membership-based sharing rule in API version 32.0 and earlier. The file name corresponds to `User.sharingRules` under the `userSharingRules` directory.

```
<?xml version="1.0" encoding="UTF-8"?>
<UserSharingRules xmlns="http://soap.sforce.com/2006/04/metadata">
    <criteriaBasedRules>
        <fullName>shareUsers2</fullName>
        <sharedTo>
            <group>Asia_Division</group>
        </sharedTo>
        <criteriaItems>
            <field>FirstName</field>
            <operation>equals</operation>
            <value>John</value>
        </criteriaItems>
        <name>shareUsers2</name>
        <userAccessLevel>Read</userAccessLevel>
    </criteriaBasedRules>
    <membershipRules>
        <fullName>shareUsers1</fullName>
        <sharedTo>
            <group>South_America_Division</group>
        </sharedTo>
        <sharedFrom>
            <group>Asia_Division</group>
        </sharedFrom>
        <name>shareUsers1</name>
        <userAccessLevel>Read</userAccessLevel>
    </membershipRules>
</UserSharingRules>
```

The following shows a sample `package.xml` file.

```
<?xml version="1.0" encoding="UTF-8"?>
<Package xmlns="http://soap.sforce.com/2006/04/metadata">
    <types>
        <members>ObjA__c.*</members>
        <name>SharingCriteriaRule</name>
    </types>
    <types>
        <members>ObjA__c.*</members>
        <name>SharingOwnerRule</name>
    </types>
    <version>48.0</version>
</Package>
```

Wildcard Support in the Manifest File

This metadata type supports the wildcard character `*` (asterisk) in the `package.xml` manifest file. For information about using the manifest file, see Deploying and Retrieving Metadata with the Zip File.

IN THIS SECTION:

1. BaseSharingRule

   This component is removed as of API version 33.0 and is available in earlier versions only. Use SharingBaseRule instead. Represents the base container for criteria-based and owner-based sharing rules.

2. CriteriaBasedSharingRule

   This component is removed as of API version 33.0 and is available in earlier versions only. Use SharingRules instead. Represents a criteria-based sharing rule. CriteriaBasedSharingRule enables you to share records based on specific criteria.

3. OwnerSharingRule

   Represents an ownership-based sharing rule. OwnerSharingRule enables you to share records owned by a set of users with another set, using rules that specify the access level of the target user group. This component is removed as of API version 33.0 and is available in earlier versions only.

BaseSharingRule

This component is removed as of API version 33.0 and is available in earlier versions only. Use SharingBaseRule instead. Represents the base container for criteria-based and owner-based sharing rules.

This type extends the Metadata metadata type and inherits its `fullName` field.

Note: You can't create a BaseSharingRule component directly. Use the components under the CriteriaBasedSharingRule or OwnerSharingRule metadata types instead.

Version

BaseSharingRule components are available in API version 24.0 and later.

Fields

For more information on these fields, see "Sharing Settings" in the Salesforce online help.

Field	Field Type	Description
`sharedTo`	SharedTo	Required. Specifies who the record should be shared with.
`fullName`	string	The unique identifier for API access. The `fullName` can contain only underscores and alphanumeric characters. It must be unique, begin with a letter, not include spaces, not end with an underscore, and not contain two consecutive underscores. This field is inherited from the Metadata component.

Wildcard Support in the Manifest File

This metadata type supports the wildcard character `*` (asterisk) in the `package.xml` manifest file. For information about using the manifest file, see Deploying and Retrieving Metadata with the Zip File.

CriteriaBasedSharingRule

This component is removed as of API version 33.0 and is available in earlier versions only. Use SharingRules instead. Represents a criteria-based sharing rule. CriteriaBasedSharingRule enables you to share records based on specific criteria.

It extends the BaseSharingRule metadata type and inherits its `sharedTo` field. For more information, see “Criteria-Based Sharing Rules Overview” in the Salesforce online help.

Note: You can’t create a CrteriaBasedSharingRule component directly. Use the child components instead.

Declarative Metadata File Suffix and Directory Location

CriteriaBasedSharingRule components are stored within the `SharingRules` component in the `criteriaBasedRules` field.

Version

CriteriaBasedSharingRule components are available in API version 24.0 and later.

Special Access Rules

As of Spring ’20 and later, only users with the View Setup and Configuration permission can access this object, and only users with the Manage Sharing permission can edit this object.

Fields

The following information assumes that you are familiar with implementing sharing rules for standard objects and custom objects. For more information on these fields, see “Sharing Settings” in the Salesforce online help.

Field	Field Type	Description
criteriaItems	FilterItem[]	List that represents the criteria for the sharing rule. The possible values are: - field - operation - value

AccountCriteriaBasedSharingRule

Represents a criteria-based sharing rule for accounts. It extends the CriteriaBasedSharingRule metadata type and inherits its criteriaItems field.

AccountCriteriaBasedSharingRule is used by the criteriaBasedRules field in AccountSharingRules.

Field	Field Type	Description
accountAccessLevel	ShareAccessLevelNoNone (enumeration of type string)	Required. A value that represents the level of access that the user or group has to the account. The possible values are: - Read - Edit - All
booleanFilter	string	Represents the filter logic of the sharing rule.
caseAccessLevel	ShareAccessLevelNoAll (enumeration of type string)	Required. A value that represents the level of access that the user or group has to cases associated with the account. The possible values are: - None - Read - Edit
contactAccessLevel	ShareAccessLevelNoAll (enumeration of type string)	Required. A value that represents the level of access that the user or group has to contacts associated with the account. The possible values are: - None - Read - Edit
description	string	Represents the description of the sharing rule. Maximum of 1000 characters. This field is available in API version 29.0 and later.
name	string	Required. Name for the sharing rule. Corresponds to **Label** in the user interface.

Field	Field Type	Description
opportunityAccessLevel	ShareAccessLevelNoAll (enumeration of type string)	Required. A value that represents the level of access that a target group is granted for any associated opportunity. The possible values are: - None - Read - Edit

CampaignCriteriaBasedSharingRule

Represents a criteria-based sharing rule for campaigns. It extends the CriteriaBasedSharingRule metadata type and inherits its criteriaItems field.

CampaignCriteriaBasedSharingRule is used by the criteriaBasedRules field in CampaignSharingRules.

Field	Field Type	Description
booleanFilter	string	Represents the filter logic of the sharing rule.
description	string	Represents the description of the sharing rule. Maximum of 1000 characters. This field is available in API version 29.0 and later.
campaignAccessLevel	ShareAccessLevelNoNone (enumeration of type string)	Required. A value that represents the level of access that a target group is granted for a campaign. The possible values are: - Read - Edit - All
name	string	Required. Name for the sharing rule. Corresponds to **Label** in the user interface.

CaseCriteriaBasedSharingRule

Represents a criteria-based sharing rule for cases. It extends the CriteriaBasedSharingRule metadata type and inherits its criteriaItems field.

CaseCriteriaBasedSharingRule is used by the criteriaBasedRules field in CaseSharingRules.

Field	Field Type	Description
booleanFilter	string	Represents the filter logic of the sharing rule.
description	string	Represents the description of the sharing rule. Maximum of 1000 characters. This field is available in API version 29.0 and later.

Field	Field Type	Description
caseAccessLevel	ShareAccessLevelReadEdit (enumeration of type string)	Required. A value that represents the level of access being granted for a case. The possible values are: • Read • Edit
name	string	Required. Name for the sharing rule. Corresponds to **Label** in the user interface.

ContactCriteriaBasedSharingRule

Represents a criteria-based sharing rule for contacts. It extends the CriteriaBasedSharingRule metadata type and inherits its criteriaItems field.

ContactCriteriaBasedSharingRule is used by the criteriaBasedRules field in ContactSharingRules.

Field	Field Type	Description
booleanFilter	string	Represents the filter logic of the sharing rule.
description	string	Represents the description of the sharing rule. Maximum of 1000 characters. This field is available in API version 29.0 and later.
contactAccessLevel	ShareAccessLevelReadEdit (enumeration of type string)	Required. A value that represents the level of access being granted to the target group, role, or user for a contact. The possible values are: • Read • Edit
name	string	Required. Name for the sharing rule. Corresponds to **Label** in the user interface.

LeadCriteriaBasedSharingRule

Represents a criteria-based sharing rule for leads. It extends the CriteriaBasedSharingRule metadata type and inherits its criteriaItems field.

LeadCriteriaBasedSharingRule is used by the criteriaBasedRules field in LeadSharingRules.

Field	Field Type	Description
booleanFilter	string	Represents the filter logic of the sharing rule.
description	string	Represents the description of the sharing rule. Maximum of 1000 characters. This field is available in API version 29.0 and later.

Field	Field Type	Description
leadAccessLevel	ShareAccessLevelReadEdit (enumeration of type string)	Required. A value that represents the level of allowed access. The possible values are: - Read - Edit
name	string	Required. Name for the sharing rule. Corresponds to **Label** in the user interface.

OpportunityCriteriaBasedSharingRule

Represents a criteria-based sharing rule for opportunities. It extends the CriteriaBasedSharingRule metadata type and inherits its criteriaItems field.

OpportunityCriteriaBasedSharingRule is used by the criteriaBasedRules field in OpportunitySharingRules.

Field	Field Type	Description
booleanFilter	string	Represents the filter logic of the sharing rule.
description	string	Represents the description of the sharing rule. Maximum of 1000 characters. This field is available in API version 29.0 and later.
opportunityAccessLevel	ShareAccessLevelReadEdit (enumeration of type string)	Required. A value that represents the level of allowed access. The possible values are: - Read - Edit
name	string	Required. Name for the sharing rule. Corresponds to **Label** in the user interface.

CustomObjectCriteriaBasedSharingRule

Represents a criteria-based sharing rule for custom objects. It extends the CriteriaBasedSharingRule metadata type and inherits its criteriaItems field.

CustomObjectCriteriaBasedSharingRule is used by the criteriaBasedRules field in CustomObjectSharingRules.

Field	Field Type	Description
accessLevel	string	Required. A value that represents the type of allowed sharing. The possible values are: - Read - Edit - All

Field	Field Type	Description
`booleanFilter`	string	Represents the filter logic of the sharing rule.
`description`	string	Represents the description of the sharing rule. Maximum of 1000 characters. This field is available in API version 29.0 and later.
`name`	string	Required. Name for the sharing rule. Corresponds to **Label** in the user interface.

UserCriteriaBasedSharingRule

Represents a criteria-based sharing rule for users. It extends the CriteriaBasedSharingRule metadata type and inherits its `criteriaItems` field.

UserCriteriaBasedSharingRule is used by the `criteriaBasedRules` field in UserSharingRules.

Field	Field Type	Description
`booleanFilter`	string	Represents the filter logic of the sharing rule.
`description`	string	Represents the description of the sharing rule. Maximum of 1000 characters. This field is available in API version 29.0 and later.
`name`	string	Required. Name for the sharing rule. Corresponds to **Label** in the user interface.
`userAccessLevel`	`ShareAccessLevelReadEdit` (enumeration of type string)	Required. A value that represents the type of allowed sharing. The possible values are: • `Read` • `Edit`

Declarative Metadata Sample Definition

The following is the definition of two owner-based sharing rules and one criteria-based sharing rule containing two criteria items. The file name corresponds to the Account.sharingRules file under the accountSharingRules directory.

```
<?xml version="1.0" encoding="UTF-8"?>
<AccountSharingRules xmlns="http://soap.sforce.com/2006/04/metadata">
  <ownerRules>
    <fullName>G1Dev_G2New</fullName>
    <sharedTo>
      <group>G2New</group>
    </sharedTo>
    <sharedFrom>
      <group>G1Dev</group>
    </sharedFrom>
    <accountAccessLevel>Read</accountAccessLevel>
    <caseAccessLevel>None</caseAccessLevel>
    <contactAccessLevel>Read</contactAccessLevel>
  </ownerRules>
    <fullName>G2New_R1New</fullName>
    <sharedTo>
      <roleAndSubordinates>R1New</roleAndSubordinates>
    </sharedTo>
    <sharedFrom>
      <group>G2New</group>
    </sharedFrom>
    <accountAccessLevel>Edit</accountAccessLevel>
    <caseAccessLevel>Read</caseAccessLevel>
    <contactAccessLevel>Edit</contactAccessLevel>
    <name>G2New_R1New</name>
    <opportunityAccessLevel>None</opportunityAccessLevel>
  </ownerRules>
  <criteriaBasedRules>
    <fullName>AccountCriteria</fullName>
    <sharedTo>
      <group>G1</group>
    </sharedTo>
    <criteriaItems>
      <field>BillingCity</field>
      <operation>equals</operation>
      <value>San Francisco</value>
    </criteriaItems>
    <criteriaItems>
      <field>MyChkBox__c</field>
      <operation>notEqual</operation>
      <value>False</value>
    </criteriaItems>
    <accountAccessLevel>Read</accountAccessLevel>
    <booleanFilter>1 OR 2</booleanFilter>
    <caseAccessLevel>None</caseAccessLevel>
    <contactAccessLevel>Read</contactAccessLevel>
    <name>AccountCriteria</name>
    <opportunityAccessLevel>None</opportunityAccessLevel>
  </criteriaBasedRules>
</AccountSharingRules>
```

Wildcard Support in the Manifest File

This metadata type supports the wildcard character `*` (asterisk) in the `package.xml` manifest file. For information about using the manifest file, see Deploying and Retrieving Metadata with the Zip File.

OwnerSharingRule

Represents an ownership-based sharing rule. OwnerSharingRule enables you to share records owned by a set of users with another set, using rules that specify the access level of the target user group. This component is removed as of API version 33.0 and is available in earlier versions only.

OwnerSharingRule extends the BaseSharingRule metadata type and inherits its SharedTo field. For more information, see “Sharing Rules” in the Salesforce online help.

Note: You can't create a OwnerSharingRule component directly. Use the child components instead.

Declarative Metadata File Suffix and Directory Location

OwnerSharingRules components are stored within the `SharingRules` component in the `ownerRules` field.

Version

OwnerSharingRules components are available in API version 24.0 and later.

Special Access Rules

As of Spring '20 and later, only users with the View Setup and Configuration permission can access this object, and only users with the Manage Sharing permission can edit this object.

Fields

The following information assumes that you are familiar with implementing sharing rules for standard objects and custom objects. For more information on these fields, see "Sharing Settings" in the Salesforce online help.

Field	Field Type	Description
`sharedFrom`	SharedTo	Required. Specifies the record owners.
`sharedTo`	SharedTo	Required. Specifies who the record should be shared with.
`fullName`	string	The unique identifier for API access. The `fullName` can contain only underscores and alphanumeric characters. It must be unique, begin with a letter, not include spaces, not end with an underscore, and not contain two consecutive underscores. This field is inherited from the Metadata component.

AccountOwnerSharingRule

Represents a sharing rule for an account with users other than the owner. It extends the OwnerSharingRule metadata type and inherits its `fullName`, `sharedFrom`, and `sharedTo` fields.

AccountOwnerSharingRule is used by the `ownerRules` field in AccountSharingRules.

Field	Field Type	Description
`accountAccessLevel`	`ShareAccessLevelNoNone` (enumeration of type string)	Required. A value that represents the level of access that a group or role has to the account. The possible values are: • `Read` • `Edit` • `All`

Field	Field Type	Description
caseAccessLevel	ShareAccessLevelNoAll (enumeration of type string)	Required. A value that represents the level of access that a group or role has to cases associated with the account. The possible values are: • None • Read • Edit
contactAccessLevel	ShareAccessLevelNoAll (enumeration of type string)	Required. A value that represents the level of access that a group or role has to contacts associated with the account. The possible values are: • None • Read • Edit
description	string	Represents the description of the sharing rule. Maximum of 1000 characters. This field is available in API version 29.0 and later.
name	string	Required. Name for the sharing rule. Corresponds to **Label** in the user interface.
opportunityAccessLevel	ShareAccessLevelNoAll (enumeration of type string)	Required. A value that represents the level of access that a group or role is granted for any associated opportunity. The possible values are: • None • Read • Edit

CampaignOwnerSharingRule

Represents a sharing rule for a campaign with users other than the owner. It extends the OwnerSharingRule metadata type and inherits its fullName, sharedFrom, and sharedTo fields.

CampaignOwnerSharingRule is used by the ownerRules field in CampaignSharingRules.

Field	Field Type	Description
campaignAccessLevel	ShareAccessLevelNoNone (enumeration of type string)	A value that represents the level of access that a group or role is granted for a campaign. The possible values are: • Read • Edit • All

Field	Field Type	Description
`description`	string	Represents the description of the sharing rule. Maximum of 1000 characters. This field is available in API version 29.0 and later.
`name`	string	Name for the sharing rule. Corresponds to **Label** in the user interface.

CaseOwnerSharingRule

Represents a sharing rule for a case with users other than the owner. It extends the OwnerSharingRule metadata type and inherits its `fullName`, `sharedFrom`, and `sharedTo` fields.

CaseOwnerSharingRule is used by the `ownerRules` field in CaseSharingRules. All the following fields are required.

Field	Field Type	Description
`caseAccessLevel`	`ShareAccessLevelReadEdit` (enumeration of type string)	Required. A value that represents the level of access that a group or role is granted for a case. The possible values are: • `Read` • `Edit`
`description`	string	Represents the description of the sharing rule. Maximum of 1000 characters. This field is available in API version 29.0 and later.
`name`	string	Required. Name for the sharing rule. Corresponds to **Label** in the user interface.

ContactOwnerSharingRule

Represents a sharing rule for a contact with users other than the owner. It extends the OwnerSharingRule metadata type and inherits its `fullName`, `sharedFrom`, and `sharedTo` fields.

ContactOwnerSharingRule is used by the `ownerRules` field in ContactSharingRules.

Field	Field Type	Description
`contactAccessLevel`	`ShareAccessLevelReadEdit` (enumeration of type string)	Required. A value that represents the level of access that a group or role is granted for a contact. The possible values are: • `Read` • `Edit`
`description`	string	Represents the description of the sharing rule. Maximum of 1000 characters. This field is available in API version 29.0 and later.

Field	Field Type	Description
name	string	Required. Name for the sharing rule. Corresponds to **Label** in the user interface.

LeadOwnerSharingRule

Represents a sharing rule for a lead with users other than the owner. It extends the OwnerSharingRule metadata type and inherits its `fullName`, `sharedFrom`, and `sharedTo` fields.

LeadOwnerSharingRule is used by the `ownerRules` field in LeadSharingRules.

Field	Field Type	Description
leadAccessLevel	ShareAccessLevelReadEdit (enumeration of type string)	Required. A value that represents the level of access that a group or role is granted for a lead. The possible values are: • Read • Edit
description	string	Represents the description of the sharing rule. Maximum of 1000 characters. This field is available in API version 29.0 and later.
name	string	Required. Required. Name for the sharing rule. Corresponds to **Label** in the user interface.

OpportunityOwnerSharingRule

Represents a sharing rule for an opportunity with users other than the owner. It extends the OwnerSharingRule metadata type and inherits its `fullName`, `sharedFrom`, and `sharedTo` fields.

OpportunityOwnerSharingRule is used by the `ownerRules` field in OpportunitySharingRules.

Field	Field Type	Description
name	string	Required. Name for the sharing rule. Corresponds to **Label** in the user interface.
description	string	Represents the description of the sharing rule. Maximum of 1000 characters. This field is available in API version 29.0 and later.
opportunityAccessLevel	ShareAccessLevelReadEdit (enumeration of type string)	Required. A value that represents the level of access that a group or role is granted for an opportunity. The possible values are: • Read • Edit

AccountTerritorySharingRule

Represents a rule for sharing an account within a territory. It extends the OwnerSharingRule metadata type and inherits its `fullName`, `sharedFrom`, and `sharedTo` fields.

AccountTerritorySharingRule is used by the `ownerRules` field in AccountTerritorySharingRules.

Field	Field Type	Description
`accountAccessLevel`	`ShareAccessLevelNoNone` (enumeration of type string)	Required. A value that represents the level of access that a Territory or TerritoryAndSubordinates group is granted for an account territory. The possible values are: - `Read` - `Edit` - `All`
`caseAccessLevel`	`ShareAccessLevelNoAll` (enumeration of type string)	Required. A value that represents the level of access that a Territory or TerritoryAndSubordinates group is granted for all child cases to an account. The possible values are: - `None` - `Read` - `Edit`
`contactAccessLevel`	`ShareAccessLevelNoAll` (enumeration of type string)	Required. A value that represents the level of access that a Territory or TerritoryAndSubordinates group is granted for all related contacts on an account. The possible values are: - `None` - `Read` - `Edit`
`description`	string	Represents the description of the sharing rule. Maximum of 1000 characters. This field is available in API version 29.0 and later.
`name`	string	Required. Name for the sharing rule. Corresponds to **Label** in the user interface.
`opportunityAccessLevel`	`ShareAccessLevelNoAll` (enumeration of type string)	Required. A value that represents the level of access that a Territory or TerritoryAndSubordinates group is granted for all opportunities associated with an account. The possible values are: - `None` - `Read` - `Edit`

CustomObjectOwnerSharingRule

Represents a sharing rule for custom objects. It extends the OwnerSharingRule metadata type and inherits its fullName, sharedFrom, and sharedTo fields.

CustomObjectOwnerSharingRule is used by the ownerRules field in CustomObjectSharingRules.

Field	Field Type	Description
accessLevel	string	Required. A value that represents the level of access that a group or role is granted to a custom object. The possible values are: - Read - Edit - All
description	string	Represents the description of the sharing rule. Maximum of 1000 characters. This field is available in API version 29.0 and later.
name	string	Required. Name for the sharing rule. Corresponds to **Label** in the user interface.

UserMembershipSharingRule

Represents a sharing rule to share members of a group with another group of users. It extends the OwnerSharingRule metadata type and inherits its fullName, sharedFrom, and sharedTo fields.

UserMembershipSharingRule is used by the ownerRules field in UserSharingRules on page 1043.

Field	Field Type	Description
description	string	Represents the description of the sharing rule. Maximum of 1000 characters. This field is available in API version 29.0 and later.
name	string	Required. Name for the sharing rule. Corresponds to **Label** in the user interface.
userAccessLevel	ShareAccessLevelReadEdit (enumeration of type string)	Required. A value that represents the level of access that a group or role is granted for a user. The possible values are: - Read - Edit

Wildcard Support in the Manifest File

This metadata type supports the wildcard character * (asterisk) in the package.xml manifest file. For information about using the manifest file, see Deploying and Retrieving Metadata with the Zip File.

SharingSet

Represents a sharing set. A sharing set defines an access mapping that grants portal or community users access to objects that are associated with their accounts or contacts. This type extends the Metadata metadata type and inherits its `fullName` field.

For example, you can grant portal or community users access to all cases related to their account record. Similarly, you can grant portal or community users access to all cases related to a parent account that is identified on the user's account record. For more information, see "Give Access to Records Using Sharing Sets" in the Salesforce Help.

File Suffix and Directory Location

SharingSet components have the suffix `.sharingSet` and are stored in the `sharingSets` folder.

Version

SharingSet components are available in API version 30.0 and later.

Special Access Rules

As of Spring '20 and later, only users with the View Setup and Configuration permission can access this object, and only users with the Manage Sharing permission can edit this object. To create or update sharing sets, you need the Customize Application permission.

Sharing sets are available with these licenses.

- Authenticated Website
- Customer Community Login
- Customer Community Plus
- Partner Community
- Customer Community User
- High Volume Customer Portal
- High Volume Portal
- Overage Authenticated Website User
- Overage High Volume Customer Portal User

Fields

Field Name	Field Type	Description
`accessMappings`	AccessMapping[]	A list of access mappings on a sharing set.
`description`	string	The sharing set description. Limit: 255 characters.
`name`	string	Required. The unique identifier for API access. Corresponds to **Sharing Set Name** on the user interface.
`profiles`	string[]	The profiles of users that are granted access to the target objects. Profiles must be associated with a license that can use sharing sets. See Special Access Rules for more information.

AccessMapping

AccessMapping represents an access mapping in the sharing set, which grants access to a target object by looking up to an account or contact associated with the user.

You can grant portal users access to a target object, or to both a target object and its associated objects, such as an account and its contacts and cases.

Field Name	Field Type	Description
`accessLevel`	string	Required. The target object access level granted to the portal user. Valid values are: - `Read` - `Edit`
`objectField`	string	Required. A lookup to the target object, which supports standard or custom fields, or an ID. For accounts or cases associated with entitlements, use `Entitlement.Account` or `Entitlement.Case`.
`object`	string	Required. The target object to which the portal user is gaining access, and refers to one of the following: - `Account` - `Campaign` - `Contact` - `Case` - Custom Objects (for example, `ObjA__c`) - `Opportunity` - `Order` - `ServiceContract` - `User` - `WorkOrder` Portal users gain access to all order entitlements and order items under an account to which they have access.
`userField`	string	Required. The user's lookup to an account, contact, or a standard or custom field derived from an account or contact. Either the user or the user's manager can be used in the lookup. Valid values are: - `Account` - `Account.`***`Field`*** - `Contact` - `Contact.`***`Field`*** - `Contact.RelatedAccount` - `Manager.Account` - `Manager.Contact` ***`Field`*** refers to a standard or custom field based on an account or contact.

Declarative Metadata Sample Definition

The following is an example of a SharingSet component that grants users access to all contacts whose `ReportsTo` fields match the users' contacts.

```
<?xml version="1.0" encoding="UTF-8"?>
<SharingSet xmlns="http://soap.sforce.com/2006/04/metadata">
  <accessMappings>
    <accessLevel>Read</accessLevel>
    <objectField>ReportsTo</objectField>
    <object>Contact</object>
    <userField>Contact</userField>
  </accessMappings>
  <description>User Access Mapping</description>
  <name>User</name>
  <profiles>customer community user</profiles>
</SharingSet>
```

The following is an example of a SharingSet component that grants users access to all cases that are related to an entitlement, which is associated with the user's account.

```
<?xml version="1.0" encoding="UTF-8"?>
<SharingSet xmlns="http://soap.sforce.com/2006/04/metadata">
  <name>Case</name>
   <accessMappings>
    <accessLevel>Edit</accessLevel>
    <objectField>Entitlement.Account</objectField>
    <object>Case</object>
    <userField>Account</userField>
  </accessMappings>
</SharingSet>
```

The following is an example of a SharingSet component with a list of access mappings.

```
<?xml version="1.0" encoding="UTF-8"?>
<SharingSet xmlns="http://soap.sforce.com/2006/04/metadata">
  <description>This is a basic sharing set with several access mappings.</description>
  <name>Basic</name>
  <profiles>customer community user</profiles>
  <accessMappings>
    <accessLevel>Read</accessLevel>
    <objectField>Id</objectField>
    <object>Account</object>
    <userField>Account</userField>
  </accessMappings>
  <accessMappings>
    <accessLevel>Edit</accessLevel>
    <objectField>Account</objectField>
    <object>Contact</object>
    <userField>Account</userField>
  </accessMappings>
  <accessMappings>
    <accessLevel>Edit</accessLevel>
    <objectField>Contact</objectField>
    <object>Case</object>
    <userField>Contact</userField>
  </accessMappings>
  <accessMappings>
    <accessLevel>Read</accessLevel>
    <objectField>AccountLookup__c</objectField>
    <object>HVPUAccessible__c</object>
    <userField>Account</userField>
  </accessMappings>
</SharingSet>
```

The following is an example `package.xml` that references the previous definition.

```
<Package xmlns="http://soap.sforce.com/2006/04/metadata">
  <fullName>SharingSetBasic</fullName>
  <types>
    <members>HVPUAccessible__c.AccountLookup__c</members>
    <members>HVPUAccessible__c.ContactLookup__c</members>
    <name>CustomField</name>
  </types>
  <types>
    <members>HVPUAccessible__c</members>
    <name>CustomObject</name>
  </types>
  <types>
    <members>Basic</members>
    <name>SharingSet</name>
  </types>
  <version>30.0</version>
</Package>
```

Wildcard Support in the Manifest File

This metadata type supports the wildcard character `*` (asterisk) in the `package.xml` manifest file. For information about using the manifest file, see Deploying and Retrieving Metadata with the Zip File.

SiteDotCom

Represents a site for deployment.

SiteDocCom extends the MetadataWithContent type and inherits its `fullName` and `content` fields.

Declarative Metadata File Suffix and Directory Location

SiteDotCom components are stored in the `siteDotComSites` directory of the corresponding package directory.

The file name for the metadata `.xml` file is `[sitename]1.site-meta.xml`. The file name for the site file is `[sitename]1.site`.

When a Lightning community is created, two sites are actually made behind the scenes: CustomSite (of type ChatterNetwork) and SiteDotComSite (of type ChatterNetworkPicasso). These are named, respectively, *<community_name>* and *<community_name>1*. The corresponding MDAPI file names are `<community_name>.site-meta.xml` and `<community_name>1.site-meta.xml`. 1 is appended to the SiteDotComSite type to keep the name unique from the corresponding CustomSite site.

Note: There is a file size limitation when using the Metadata API to deploy a site from sandbox to production. The assets in the `.site` file can't be larger than 40 MB. The site gets created, but the assets show in the new site as broken. To fix the assets, export the assets from the sandbox environment separately and then import them into your new site.

Version

`SiteDotCom` components are available in API version 30.0 and later.

Fields

Field	Field Type	Description
`label`	string	The name of the site you are deploying.
`siteType`	(enumeration of type string)	Required. Identifies whether the site is a `ChatterNetworkPicasso` site for Salesforce Communities sites, or a `Siteforce` site for Site.com sites.

Declarative Metadata Sample Definition

Sample XML definitions for SiteDotCom are shown below.

```
<?xml version="1.0" encoding="UTF-8"?>
<SiteDotCom xmlns="http://soap.sforce.com/2006/04/metadata">
    <label>testsite</label>
    <siteType>Siteforce</siteType>
</SiteDotCom>


<?xml version="1.0" encoding="UTF-8"?>
<SiteDotCom xmlns="http://soap.sforce.com/2006/04/metadata">
    <label>testCommunity</label>
    <siteType>ChatterNetworkPicasso</siteType>
</SiteDotCom>
```

Wildcard Support in the Manifest File

This metadata type supports the wildcard character `*` (asterisk) in the `package.xml` manifest file. For information about using the manifest file, see Deploying and Retrieving Metadata with the Zip File.

Skill

Represents the settings for a skill used for field service or to route chats to agents in Chat, such as the name of the skill and which agents the skills are assigned to.

This type extends the Metadata metadata type and inherits its `fullName` field.

File Suffix and Directory Location

Skill values are stored in the `<developer_name>.skill` file in the `skills` directory.

Version

Skill is available in API version 28.0 and later.

Fields

Field Name	Field Type	Description
`assignments`	SkillAssignments	Specifies how skills are assigned to Chat users. Skills can be assigned to sets of users or sets of profiles.
`description`	string	Specifies the description of the skill. This field is available in API version 38.0 and later.
`label`	string	Specifies the name of the skill.

SkillAssignments

Represents which users and user profiles to whom specific skills are assigned.

Fields

Field Name	Field Type	Description
`profiles`	SkillProfileAssignments	Specifies the profiles that are associated with a specific skill.
`users`	SkillUserAssignments	Specifies the users that are associated with a specific skill.

SkillProfileAssignments

Represents the profiles that are associated with a specific skill.

Fields

Field Name	Field Type	Description
`profile`	string	Specifies the custom name of the profile associated with a specific skill.

SkillUserAssignments

Represents the users that are associated with a specific skill.

Fields

Field Name	Field Type	Description
`user`	string	Specifies the username of the user associated with a specific skill.

Declarative Metadata Sample Definition

This is a sample of a `skill` file.

```
<?xml version="1.0" encoding="UTF-8"?>
<Skill xmlns="http://soap.sforce.com/2006/04/metadata">
    <label>My Skill 1</label>
    <assignments>
        <profiles>
            <profile>LiveAgentOperator</profile>
            <profile>LiveAgentSupervisor</profile>
        </profiles>
        <users>
            <user>jdoe@acme.com</user>
        </users>
    </assignments>
</Skill>
```

Wildcard Support in the Manifest File

This metadata type supports the wildcard character `*` (asterisk) in the `package.xml` manifest file. For information about using the manifest file, see Deploying and Retrieving Metadata with the Zip File.

StandardValueSet

Represents the set of values in a standard picklist field. This type extends the Metadata metadata type and inherits its `fullName` field.

File Suffix and Directory Location

StandardValueSet components have the suffix `.standardValueSet` and are stored in the `standardValueSets` folder.

Version

StandardValueSet components are available in API version 38.0 and later.

Fields

Field Name	Field Type	Description
`groupingStringEnum`	string	Groups picklist and enumerated values. For example, for the picklist values of the `Status` field on the Service Appointment object, `Done` and `Finished` can both have a grouping string of `Completed`. Available in API version 41.0 and later.
`sorted`	boolean	Required. Indicates whether a global value set is sorted in alphabetical order. By default, this value is `false`.
`standardValue`	StandardValue[]	Defines each value in a standard picklist's value set. The `groupingString` value is available in API version 38.0 and later Note: When setting `standardValue` on Record Types, including person account record types, new picklist values loaded into your organization through the Metadata API do not display in the picklist UI by default. For users to see the new values, go to the Record Types list for the object containing the picklist field, click **Edit** and add the new value to the Selected Fields list.

Declarative Metadata Sample Definition

The following example shows a StandardValueSet component that's defined as the Stage standard picklist on a customized opportunity object.

```
<?xml version="1.0" encoding="UTF-8"?>
<StandardValueSet xmlns="http://soap.sforce.com/2006/04/metadata">
    <fullName>OpportunityStage</fullName> <!-- Enum name -->
    <standardValue>
       <fullName>Closed Abandoned</fullName>
    </standardValue>
    <standardValue>
       <fullName>Closed Won</fullName>
    </standardValue>
    <standardValue>
       <fullName>Closed Lost</fullName>
    </standardValue>
</StandardValueSet>

<CustomObject>
    <fullName>Opportunity</fullName>
    <fields>
       <fullName>StageName</fullName> <!-- field name -->
       <label>Stage</label>
       <type>Picklist</type>
    </fields>
    <label>ObjectWithValueSet</label>
    <pluralLabel>ObjectWithValueSet</pluralLabel>
    <sharingModel>ReadWrite</sharingModel>
</CustomObject>
```

For a list of standard value set names for standard picklists, see StandardValueSet Names and Standard Picklist Fields.

Wildcard Support in the Manifest File

This metadata type doesn't support the wildcard character `*` (asterisk) in the `package.xml` manifest file. For information about using the manifest file, see Deploying and Retrieving Metadata with the Zip File.

StandardValueSetTranslation

Contains details for a standard picklist translation. It returns a translated standard value set.This type extends the Metadata metadata type and inherits its `fullName` field.

File Suffix and Directory Location

StandardValueSetTranslation components have the suffix `.standardValueSetTranslation` and are stored in the `standardValueSetTranslations` folder.

Translations are stored in a file with a format of `ValueSetName-lang.standardValueSetTranslation`, where *`ValueSetName`* is the global value set's name, and lang is the translation language.

Version

StandardValueSetTranslation components are available in API version 38.0 and later.

Fields

Field	Field Type	Description
`valueTranslation`	ValueTranslation[]	A list of values from global value sets to be translated.

Declarative Metadata Sample Definition

The following is an example of a StandardValueSetTranslation component. When a value isn't translated, its translation becomes a comment that's paired with its masterLabel.

```
<?xml version="1.0" encoding="UTF-8"?>
<StandardValueSetTranslation xmlns="http://soap.sforce.com/2006/04/metadata">
    <valueTranslation>
        <masterLabel>Cold</masterLabel>
        <translation><!-- Cold --></translation>
    </valueTranslation>
    <valueTranslation>
        <masterLabel>Hot</masterLabel>
        <translation><!-- Hot --></translation>
    </valueTranslation>
    <valueTranslation>
        <masterLabel>Warm</masterLabel>
        <translation><!-- Warm --></translation>
    </valueTranslation>
</StandardValueSetTranslation>
```

The following is an example `package.xml` that references the StandardValueSetTranslation definition.

```
<?xml version="1.0" encoding="UTF-8"?>
<Package xmlns="http://soap.sforce.com/2006/04/metadata">
        <types>
        <members>AccountRating-fr</members>
        <name>StandardValueSetTranslation</name>
    </types>
    <version>38.0</version>
</Package>
```

Wildcard Support in the Manifest File

This metadata type supports the wildcard character `*` (asterisk) in the `package.xml` manifest file. For information about using the manifest file, see Deploying and Retrieving Metadata with the Zip File.

SEE ALSO:

Translations

StaticResource

Represents a static resource file, often a code library in a ZIP file. Static resources allow you to upload content that you can reference in a Visualforce page, including archives (such as .zip and .jar files), images, style sheets, JavaScript, and other files.

This type extends the MetadataWithContent metadata type and inherits its `content` and `fullName` fields.

File Suffix and Directory Location

The file suffix is `.resource` for the template file. The accompanying metadata file is named *`resource`*`-meta.xml`.

Static resource components are stored in the `staticresources` folder in the corresponding package directory.

Version

Static resources are available in API version 12.0 and later.

Fields

This metadata type contains the following fields:

Field Name	Field Type	Description
`cacheControl`	StaticResourceCacheControl (enumeration of type string)	Required. Indicates whether the static resource is marked with a public caching tag so that a third-party delivery client can cache the content. This is a new field in API version 14.0. The valid values are: • Private • Public
`content`	base64Binary	The static resource content. Base 64-encoded binary data. Prior to making an API call, client applications must encode the binary attachment data as base64. Upon receiving a response, client applications must decode the base64 data to binary. This conversion is usually handled for you by a SOAP client. This field is inherited from the MetadataWithContent component.
`contentType`	string	Required. The content type of the file, for example text/plain.
`description`	string	The description of the static resource.
`fullName`	string	The static resource name. The name can only contain characters, letters, and the underscore (_) character, must start with a letter, and cannot end with an underscore or contain two consecutive underscore characters. Inherited from the Metadata component, this field is not defined in the WSDL for this component. It must be specified when creating, updating, or deleting. See `create()` to see an example of this field specified for a call.

Declarative Metadata Sample Definition

```
<?xml version="1.0" encoding="UTF-8"?>
<StaticResource xmlns="http://soap.sforce.com/2006/04/metadata">
    <contentType>text/plain</contentType>
    <description>Test Resource</description>
</StaticResource>
```

Wildcard Support in the Manifest File

This metadata type supports the wildcard character `*` (asterisk) in the `package.xml` manifest file. For information about using the manifest file, see Deploying and Retrieving Metadata with the Zip File.

SynonymDictionary

Represents a set of synonym groups, which are groups of words or phrases that are treated as equivalent in users' searches. You can define synonym groups to optimize search results for acronyms, variations of product names, and other terminology unique to your organization.

Synonyms are available in Service Cloud features such as Salesforce Knowledge. This type extends the Metadata metadata type and inherits its `fullName` field.

File Suffix and Directory Location

SynonymDictionary components have the suffix `.synonymDictionary` and are stored in the `synonymDictionaries` folder.

Version

SynonymDictionary components are available in API version 29.0 and later.

Special Access Rules

Synonyms must be enabled in your organization. Only users with the "Manage Synonyms" permission can access this object.

Fields

Field Name	Field Type	Description
`groups`	SynonymGroup	The synonym groups defined in this dictionary.
`isProtected`	boolean	Indicates whether this component is protected (`true`) or not (`false`). Protected components cannot be linked to or referenced by components created in the installing organization.
`label`	string	Required. Specifies the display name of the synonym dictionary.

SynonymGroup

Represents a group of synonymous words or phrases.

Field Name	Field Type	Description
`languages`	Language	Required. Specifies the languages the synonym group applies to. If synonyms are specific to a single language, specify only that language. If the synonyms apply to multiple languages, specify multiple languages for one synonym group.
`terms`	string	Required. A word or phrase synonymous with other terms in the group. Maximum of 50 characters. Minimum of two `terms` per group. Synonym groups are symmetric, which means that if oranges and apples are defined in a synonym group, a search for *`oranges`* will return a match for *`apples`*, and vice versa for a search for *`apples`*.

Declarative Metadata Sample Definition

The following is an example of a SynonymDictionary component:

```
<?xml version="1.0" encoding="UTF-8"?>
<SynonymDictionary xmlns="http://soap.sforce.com/2006/04/metadata">
    <groups>
```

```
        <languages>en_US</languages>
        <terms>Salesforce</terms>
        <terms>salesforce.com</terms>
        <terms>The Customer Company</terms>
        <terms>SFDC</terms>
    </groups>
    <groups>
        <languages>fr</languages>
        <terms>renault</terms>
        <terms>clio</terms>
    </groups>
    <label>Sample Dictionary</label>
</SynonymDictionary>
```

The following is an example `package.xml` that references the SynonymDictionary component.

```
<?xml version="1.0" encoding="UTF-8"?>
<Package xmlns="http://soap.sforce.com/2006/04/metadata">
    <types>
        <members>Sample Dictionary</members>
        <name>SynonymDictionary</name>
    </types>
    <version>48.0</version>
</Package>
```

Usage

If you have existing synonym groups defined before API version 29.0, your existing groups are associated with a default dictionary called `_Default`.

If you have a set of synonyms that require frequent updates, we recommend assigning the synonym group or groups to a dedicated dictionary with a small number of groups. Each time you deploy an existing dictionary, all of its synonym groups are overwritten. We don't support deploying updates to only a single synonym group within a dictionary.

Wildcard Support in the Manifest File

This metadata type supports the wildcard character `*` (asterisk) in the `package.xml` manifest file. For information about using the manifest file, see Deploying and Retrieving Metadata with the Zip File.

Territory

Represents a territory in your organization.

Declarative Metadata File Suffix and Directory Location

The file suffix for territory components is `.territory` and components are stored in the `territories` directory of the corresponding package directory.

Version

Territory components are available in API version 24.0 and later.

Fields

This metadata type extends to subtype RoleOrTerritory.

Field Name	Field Type	Description
accountAccessLevel	string	Specifies whether users in this territory can access accounts that are assigned to this territory and are otherwise inaccessible. Valid values are: • Read • Edit • All If your organization's sharing model for accounts is Public Read/Write, valid values are only Edit and All. If no value is set for this field, this field value uses the default access level that is specified in the Manage Territory page in Setup. This field is available in API version 31.0 and later.
fullName	string	The unique identifier for API access. The fullName can contain only underscores and alphanumeric characters. It must be unique, begin with a letter, not include spaces, not end with an underscore, and not contain two consecutive underscores. This field is inherited from the Metadata component. Corresponds to **Territory Name** in the user interface.
parentTerritory	string	The territory above this territory in the territory hierarchy.

Declarative Metadata Sample Definition

The following is the definition of a territory.

```xml
<?xml version="1.0" encoding="UTF-8"?>
<Territory xmlns="http://soap.sforce.com/2006/04/metadata">
    <accountAccessLevel>Edit</accountAccessLevel>
    <caseAccessLevel>Edit</caseAccessLevel>
    <contactAccessLevel>Edit</contactAccessLevel>
    <description>Sample Territory</description>
    <mayForecastManagerShare>false</mayForecastManagerShare>
    <name>T22name</name>
    <opportunityAccessLevel>Read</opportunityAccessLevel>
</Territory>
```

Wildcard Support in the Manifest File

This metadata type supports the wildcard character * (asterisk) in the package.xml manifest file. For information about using the manifest file, see Deploying and Retrieving Metadata with the Zip File.

Territory2

Represents the metadata associated with a sales territory in Territory Management 2.0. This type extends the Metadata metadata type and inherits its `fullName` field.Available only if Territory Management 2.0 has been enabled for your organization.

File Suffix and Directory Location

Territory2 components have the suffix `territory2` and are stored in the `territories` folder under the folder for the corresponding Territory2Model.

Version

Territory2 components are available in API version 32.0 and later.

Special Access Rules

The Territory2Model object has a `State` field in the SOAP API. States include `Planning`, `Active`, `Archived`, and a number of other states, such as `Cloning`, that indicate that a process is underway. Users who do not have the "Manage Territories" permission can access only territories that belong to the model in `Active` state. The "Manage Territories" permission is required for `deploy()` calls for all territory management entities, in addition to the "Modify All Data" permission required by Metadata API. Using `retrieve()` without the "Manage Territories" permission will return only entities that belong to a Territory2Model in `Active` state. We recommend against retrieving without the "Manage Territories" permission because the call will retrieve only partial data.

Fields

Field Name	Field Type	Description
`accountAccessLevel`	string	Specifies whether users in this territory can access accounts that are assigned to this territory and are otherwise inaccessible. Valid values are: • `Read` • `Edit` • `All` If your organization's sharing model for accounts is Public Read/Write, valid values are only `Edit` and `All`. If no value is set for this field, this field value uses the default access level that is specified in Territory2Settings as permitted by the organization's sharing settings.
`caseAccessLevel`	string	Specifies whether users in this territory can access cases that are assigned to this territory and are otherwise inaccessible. Valid values are: • `None` • `Read` • `Edit`

Field Name	Field Type	Description
		No value should be specified if your organization’s sharing model for cases/opportunities is Public Read/Write,If no value is set for this field, this field value uses the default access level that is specified in Territory2Settings as permitted by the organization’s sharing settings.
`contactAccessLevel`	string	Specifies whether users in this territory can access contacts that are assigned to this territory and are otherwise inaccessible. Valid values are: • `None` • `Read` • `Edit` No value should be specified if your organization’s sharing model for contacts is Public Read/Write or Controlled By Parent,
`customFields`	FieldValue	Values for custom fields defined on the Territory2 object and used by this territory. Their metadata is captured separately in CustomObject on page 319. Note the following: • Territory2 and Territory2Model objects do not handle values for Text Area (Long), Text Area (Rich), and text-encrypted custom fields. • Fields are referenced using their API names. Compound field types like Location will appear as their constituent column fields. For example, `nnn_Latitude__s`, `nnn_Longitude__s` where “nnn” is the field name and the suffixes are the geolocation components. • Values of required custom fields are enforced during the `deploy()` operation.
`description`	string	A description of the territory.
`name`	string	Required. The user interface label for the territory.
`opportunityAccessLevel`	string	Specifies whether users in this territory can access opportunities that are assigned to this territory and are otherwise inaccessible. Valid values are: • `None` • `Read` • `Edit` No value should be specified if your organization’s sharing model for cases/opportunities is Public Read/Write,If no value is set for this field, this field value uses the default access level that is specified in Territory2Settings as permitted by the organization’s sharing settings.
`parentTerritory`	string	The name of the territory’s parent. When you specify the parent territory, use the developer name. Do not use the “fully qualified” name. Custom fields with no values are retrieved with values of type: `<value xsi:nil="true"/>`. You can also use `<value`

Field Name	Field Type	Description
		xsi:nil="true"/> syntax to remove existing values in custom fields.
ruleAssociations	Territory2RuleAssociation	Represents an object assignment rule and its association to a territory. Use the developer name of the rule.
territory2Type	string	Required. The territory type that the territory belongs to.

FieldValue

Represents the values of custom fields on the Territory2 object. Available in API version 32.0 and later.

Field Name	Field Type	Description
name	string	Required. The user interface label for the territory.
value	any type	The value of the field, which can also be null. The field type is specified in the XML and depends on the field value.

Territory2RuleAssociation

Represents the association of an object assignment rule to a territory. Available in API version 32.0 and later.

Field Name	Field Type	Description
inherited	boolean	Required. Indicates whether the rule is inherited from a parent territory (true) or local to the current territory (false). Rule inheritance flows from the parent territory where the rule is created to the rule's descendent territories (if any) in the territory model hierarchy. A local rule is created within a single territory and affects that territory only.
ruleName	string	Required. The name of a rule associated with the territory. ruleName doesn't need to be fully qualified because Metadata API assumes that the rule belongs to the same model as the territory.

Declarative Metadata Sample Definition

The following example shows the definition of a Territory2 component.

```
<?xml version="1.0" encoding="UTF-8"?>
<Territory2 xmlns="http://soap.sforce.com/2006/04/metadata"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns:xsd="http://www.w3.org/2001/XMLSchema">
    <name>USA</name>
    <description>United States sales</description>
    <accountAccessLevel>Edit</accountAccessLevel>
    <opportunityAccessLevel>Read</opportunityAccessLevel>
    <caseAccessLevel>Edit</caseAccessLevel>
    <contactAccessLevel>Edit</contactAccessLevel>
    <parentTerritory>Worldwide_Sales</parentTerritory>
    <territory2Type>Geo</territory2Type>
    <ruleAssociations>
        <ruleName>AccRule1</name>
        <inherited>True</inherited>
     </ruleAssociations>
     <ruleAssociations>
        <ruleName>AccRule2</name>
        <inherited>False</inherited>
     </ruleAssociations>
     <customFields>
        <name>Activation_DateTime__c</name>
        <value xsi:type="xsd:dateTime">2014-07-16T05:05:00.000Z</value>
     </customFields>
     <customFields>
        <name>AutoNumber__c</name>
        <value xsi:type="xsd:string">T# 000001</value>
     </customFields>
     <customFields>
        <name>DeactivationDate__c</name>
        <value xsi:type="xsd:date">2016-07-12</value>
     </customFields>
     <customFields>
        <name>External_Id__c</name>
        <value xsi:type="xsd:string">AB2345</value>
     </customFields>
     <customFields>
        <name>ManagersPhone__c</name>
        <value xsi:nil="true"/>
     </customFields>
</Territory2>
```

The following is a `package.xml` sample. *`FY13`* and *`FY14`* represent the names of territory models and demonstrate that rules can have identical developer names within different models. A wildcard character (*) in place of the model name can be used to retrieve all rules in all models in an organization.

```
<?xml version="1.0" encoding="UTF-8"?>
<Package xmlns="http://soap.sforce.com/2006/04/metadata">
    <types>
        <members>FY13</members>
        <members>FY14</members>
        <name>Territory2Model</name>
    </types>

    <types>
        <members>FY13.USA</members>
        <members>FY13.Worldwide_Sales</members>
        <members>FY14.APAC</members>
        <members>FY14.USA</members>
        <name>Territory2</name>
    </types>
    <version>48.0</version>
</Package>
```

Usage

- Triggers defined on Territory2 will *not* fire during a `deploy()` operation.
- Territory Management 2.0 components don't support packaging or change sets and aren't supported in CRUD calls.
- For unlocked packaging, `Territory2` requires packages without a namespace.

Wildcard Support in the Manifest File

This metadata type supports the wildcard character `*` (asterisk) in the `package.xml` manifest file. For information about using the manifest file, see Deploying and Retrieving Metadata with the Zip File.

Territory2Model

Represents the metadata associated with a territory model in Territory Management 2.0. This type extends the Metadata metadata type and inherits its `fullName` field.Available only if Territory Management 2.0 has been enabled for your Salesforce org.

File Suffix and Directory Location

Territory2Model components have the suffix `territory2Model` and are stored in the `territory2Models` folder.

Version

Territory2Model components are available in API version 32.0 and later.

Special Access Rules

The Territory2Model object has a `State` field in the SOAP API. States include `Planning`, `Active`, `Archived`, and a number of other states, such as `Cloning`, that indicate that a process is underway. Users who do not have the "Manage Territories" permission can access only models in `Active` state. The "Manage Territories" permission is required for `deploy()` calls for all territory management entities, in addition to the "Modify All Data" permission required by Metadata API. Using `retrieve()` without the "Manage Territories" permission will return only entities that belong to a Territory2Model in `Active` state. We recommend against retrieving without the "Manage Territories" permission because the call will retrieve only partial data.

Fields

Field Name	Field Type	Description
`customFields`	FieldValue	Custom fields defined on the Territory2Model object and used by this model. Their metadata is captured separately. • Territory2 and Territory2Model objects do not handle values for Text Area (Long), Text Area (Rich), and text-encrypted custom fields.

Field Name	Field Type	Description
		• Fields are referenced using their API names. Compound field types like Location will appear as their constituent column fields. For example, `nnn_Latitude__s`, `nnn_Longitude__s` where “nnn” is the field name and the suffixes are the geolocation components. • Values of required custom fields are enforced during the `deploy()` operation.
`description`	string	A description for the territory model.
`name`	string	Required. The user interface label for the territory model.

Declarative Metadata Sample Definition

The following example shows the definition of a Territory2Model component.

```
<?xml version="1.0" encoding="UTF-8"?>
    <Territory2Model xmlns="http://soap.sforce.com/2006/04/metadata"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns:xsd="http://www.w3.org/2001/XMLSchema">
    <name>FY13</name>
    <description>Geographic allocation</description>
    <customFields>
        <name>Activation_DateTime__c</name>
        <value xsi:type="xsd:dateTime">2014-07-16T05:05:00.000Z</value>
    </customFields>
    <customFields>
        <name>AutoNumber__c</name>
        <value xsi:type="xsd:string">M# 000001</value>
    </customFields>
    <customFields>
        <name>DeactivationDate__c</name>
        <value xsi:type="xsd:date">2016-07-12</value>
    </customFields>
    <customFields>
        <name>External_Id__c</name>
        <value xsi:nil="true"/>
    </customFields>
</Territory2Model>
```

Usage

- The `retrieve()` call *will not* return models in these four states: `Cloning`, `Cloning Failed`, `Deleting`, and `Deletion Failed`.
- Whenever a model is created, its initial state is `Planning`. You can only do a `deploy()` operation for models in `Planning` or `Active` state. The same requirement applies to territories and rules associated with those models. For example, you might have a model in `Planning` state on a sandbox org, and a model with the same developer name in `Archived` state on your production

org. The `deploy()` operation on production will fail because that model's state is `Archived` and that state prevents changes to the model.

- Because of the state restrictions, if you have territory models in different orgs with identical developer names and you attempt a `deploy()` operation, Metadata API will attempt to create new models, but that operation will fail because of the developer name conflict. For example, you might have a model in `Planning` state on a sandbox org, and a model with the same developer name in `Archived` state on your production org. The `deploy()` operation on production will fail because that model's state is `Archived` and that state prevents changes to the model.
- If you try to delete a model that has territories, then the `delete()` call will change the model's state to `Deleting` and cascade delete all territories, rules, and user associations in the model. Deleting may take some time depending on the number of territries in the model.
- Whenever a model is created, its initial state is `Planning`. If a model with the same developer name already exists, it will already have a state, so we do not include the `State` field in Territory2.
- Territory Management 2.0 components don't support packaging or change sets and aren't supported in CRUD calls.
- Namespaces aren't supported for unlocked packages.

Wildcard Support in the Manifest File

This metadata type supports the wildcard character `*` (asterisk) in the `package.xml` manifest file. For information about using the manifest file, see Deploying and Retrieving Metadata with the Zip File.

Territory2Rule

Represents the metadata associated with a territory assignment rule associated with an object, such as Account, in Territory Management 2.0. This type extends the Metadata metadata type and inherits its `fullName` field.Available only if Territory Management 2.0 has been enabled for your Salesforce org.

File Suffix and Directory Location

Territory2Rule components have the suffix `territory2Rule` and are stored in the `rules` folder under the folder for the corresponding Territory2Model.

Version

Territory2Rule components are available in API version 32.0 and later.

Special Access

The Territory2Model object has a `State` field in the SOAP API. States include `Planning`, `Active`, `Archived`, and a number of other states, such as `Cloning`, that indicate that a process is underway. Users who do not have the "Manage Territories" permission can access only rules that belong to the model in `Active` state. The "Manage Territories" permission is required for `deploy()` calls for all territory management entities, in addition to the "Modify All Data" permission required by Metadata API. Using `retrieve()` without the "Manage Territories" permission will return only entities that belong to a Territory2Model in `Active` state. We recommend against retrieving without the "Manage Territories" permission because the call will retrieve only partial data.The SOAP API and the user interface require that a user attempting to create or edit a rules has field-level security access to the fields referenced in the rule item. This restriction is relaxed for Metadata API `deploy()` operations, as those require "Modify All Data" and "Manage Territories" permissions. "Modify All Data" is the base permission requirement for all Metadata API operations.

Fields

Field Name	Field Type	Description
`active`	boolean	Required. Indicates whether the rule is active (`true`) or inactive (`false`). Via the API, active rules run automatically when object records are created and edited. The exception is when the value of the `IsExcludedFromRealign` field on an object record is `true`, which prevents record assignment rules from evaluating that record.
`booleanFilter`	string	An advanced filter condition. For example: `(1 AND 2) OR 3`. Numbering must start at 1 and must be contiguous.
`name`	string	Required. The user interface label for the rule.
`objectType`	string	Required. The object that the rule is defined for. For API version 32.0, the only available object is Account.
`ruleItems`	Territory2RuleItem on page 1081	The items that define a rule's the selection criteria, such as `Billing State equals California`.

Territory2RuleItem

Represents the association of a rule item to a rule. Available in API version 32.0 and later.

Field Name	Field Type	Description
`field`	string	The standard or custom object field that the rule item operates on.
`operation`	FilterOperation (enumeration of type string)	The criterion to apply for the rule item. For example: *`equals`* or *`starts with`*.
`value`	string	The field value or values to evaluate. For example: if the field is `Billing ZIP/Postal Code`, a value might be `94105`.

Declarative Metadata Sample Definition

The following example shows the definition of a Territory2RuleItem component.

```
<?xml version="1.0" encoding="UTF-8"?>
<Territory2Rule xmlns="http://soap.sforce.com/2006/04/metadata">
    <label>Northern CA</label>
    <description>To capture northern CA based accounts</description>
    <objectType>Account</objectType>
    <active>True</active>
    <ruleItems>
        <field>BillingZip</field>
        <operation>contains</operation>
        <value><94105,94404,94536/value>
    </ruleItems>
    <ruleItems>
        <field>Industry</field>
        <operation>equals</operation>
        <value>IT</value>
    </ruleItems>
    <ruleItems>
        <field>someCustomField__c</field>
        <operation>greater_than</operation>
        <value>50000</value>
    </ruleItems>
    <booleanFilter>(1 OR 2) AND 3</booleanFilter>
</Territory2Rule>
```

The following is a `package.xml` sample. *`FY13`* and *`FY14`* represent names of territory models and demonstrate that rules can have *identical* developer names within *different* models. A wildcard character (*) in place of the model name can be used to retrieve all rules in all models in an org.

```

<?xml version="1.0" encoding="UTF-8"?>
<Package xmlns="http://soap.sforce.com/2006/04/metadata">
        <types>
            <members>FY13</members>
            <members>FY14</members>
            <name>Territory2Model</name>
        </types>

        <types>
            <members>FY13.AccRule1</members>
            <members>FY14.AccRule1</members>
            <name>Territory2Rule</name>
        </types>

        <version>48.0</version>
</Package>
```

Usage

- A territory rule can have up to 10 rule items.
- The sort order of rule items is implicitly derived from the position of the rule items in the XML
- Rules can’t be run via Metadata API.
- Territory Management 2.0 components don’t support packaging or change sets and aren’t supported in CRUD calls.

Wildcard Support in the Manifest File

This metadata type supports the wildcard character `*` (asterisk) in the `package.xml` manifest file. For information about using the manifest file, see Deploying and Retrieving Metadata with the Zip File.

Territory2Type

Represents the metadata for a category of territories in Territory Management 2.0. Every Territory2 must have a Territory2Type. This type extends the Metadata metadata type and inherits its `fullName` field.Available only if Enterprise Territory Management has been enabled for your Salesforce org.

File Suffix and Directory Location

Territory2Type components have the suffix `territory2Type` and are stored in the `territory2Types` folder.

Version

Territory2Type components are available in API version 32.0 and later.

Special Access Rules

Users without the “Manage Territories” permission will be able to retrieve all the Territory2Types in the org. “Manage Territories” permission is required for the `deploy()` operation, in addition to the “Modify All Data” permission required by the Metadata API.

Fields

Field Name	Field Type	Description
`description`	string	A description of the territory type.
`name`	string	Required. The user interface label for the territory type.
`priority`	int	Required. Used for Filter-Based Opportunity Territory Assignment (Pilot in Spring '15 / Metadata API version 33). Lets you specify a priority for a territory type. For opportunity assignments, the filter examines all territories assigned to the account that the opportunity is assigned to. The account-assigned territory whose territory type priority is highest is then assigned to the opportunity. The `priority` field value on each territory type must be unique. Further, if there are multiple territories with the same territory type (and therefore the same priority) assigned to the account, no territory is not assigned to the opportunity.

Declarative Metadata Sample Definition

The following example shows the definition of a Territory2Type component.

```
<?xml version="1.0" encoding="UTF-8"?>
<Territory2Type xmlns="http://soap.sforce.com/2006/04/metadata">
    <name>Geo</name>
    <description>Geographic allocation</description>
</Territory2Type>
```

Usage

Territory Management 2.0 components don’t support packaging or change sets and aren’t supported in CRUD calls.

Wildcard Support in the Manifest File

This metadata type supports the wildcard character `*` (asterisk) in the `package.xml` manifest file. For information about using the manifest file, see Deploying and Retrieving Metadata with the Zip File.

TimeSheetTemplate

Represents a template for creating time sheets in Field Service Lightning.This type extends the Metadata metadata type and inherits its `fullName` field.

File Suffix and Directory Location

TimeSheetTemplate components have the suffix `timeSheetTemplate` and are stored in the `timeSheetTemplates` folder.

Version

TimeSheetTemplate components are available in API version 46.0 and later.

Special Access Rules

Field Service Lightning must be enabled. Users must have the Customize Application and Time Sheet Template permissions.

Fields

Field Name	Field Type	Description
`active`	boolean	Required. Indicates whether the time sheet template is active (`true`) or not (`false`).
`description`	string	The time sheet template's description.
`frequency`	TimeSheetFrequency (enumeration of type string)	Required. Defines the frequency of the time sheet creation period. One of the following values: • `Daily` • `Weekly` • `EveryTwoWeeks` • `TwiceAMonth` • `Monthly`
`masterLabel`	string	Required. The name of the time sheet template.
`startDate`	date	Required. The date when the time sheet takes effect.

Field Name	Field Type	Description
timeSheetTemplateAssignments	TimeSheetTemplateAssignment	A list of profiles that the template is assigned to.
workWeekEndDay	DaysOfWeek (enumeration of type string)	Required. The end day of the template's work week. One of the following values: - Monday - Tuesday - Wednesday - Thursday - Friday - Saturday - Sunday
workWeekStartDay	DaysOfWeek (enumeration of type string)	Required. The start day of the template's work week. One of the following values: - Monday - Tuesday - Wednesday - Thursday - Friday - Saturday - Sunday

TimeSheetTemplateAssignment

Returns a quick action that's associated with an EmbeddedServiceLiveAgent setup. The quick action includes the pre-chat form fields that the embedded chat window displays and shows the order in which the fields are displayed.

Field Name	Field Type	Description
assignedTo	string	The IDs of the user profiles that a time sheet template is assigned to.

Declarative Metadata Sample Definition

The following is an example of a TimeSheetTemplate file.

```
<?xml version=“1.0” encoding=“UTF-8"?>
<TimeSheetTemplate xmlns=“http://soap.sforce.com/2006/04/metadata“>
   <active>true</active>
   <description>Time Sheet Template description</description>
   <frequency>Daily</frequency>
   <masterLabel>label</masterLabel>
   <startDate>2018-10-18</startDate>
   <timeSheetTemplateAssignments>
```

```
        <assignedTo>admin</assignedTo>
    </timeSheetTemplateAssignments>
    <timeSheetTemplateAssignments>
        <assignedTo>standard</assignedTo>
    </timeSheetTemplateAssignments>
    <workWeekEndDay>Tuesday</workWeekEndDay>
    <workWeekStartDay>Monday</workWeekStartDay>
</TimeSheetTemplate>
```

The following is an example `package.xml` that references the previous definition.

```
<?xml version=“1.0” encoding=“UTF-8"?>
<Package xmlns=“http://soap.sforce.com/2006/04/metadata“>
    <types>
        <members>*</members>
        <name>TimeSheetTemplate</name>
    </types>
    <version>46.0</version>
</Package>
```

Wildcard Support in the Manifest File

This metadata type supports the wildcard character `*` (asterisk) in the `package.xml` manifest file. For information about using the manifest file, see Deploying and Retrieving Metadata with the Zip File.

TopicsForObjects

Represents the ability to assign topics to objects or to remove topic assignments.

File Suffix and Directory Location

TopicsForObjects components have the suffix `.topicsForObjects` and are stored in the `topicsForObjects` folder of the corresponding package directory.

Version

TopicsForObjects components are available in API version 41.0 and later.

Fields

Field Name	Field Type	Description
`enableTopics`	boolean	Required. When true, indicates whether users can assign topics or remove topic assignments. When false, users can’t assign or remove topics. Upon org creation, this value is true for the following objects: • Account • Asset

Field Name	Field Type	Description
		• Campaign • Case • Contact • Content Document • Contract • Event • Lead • Opportunity • Order • Solution • Task For all remaining standard objects and custom objects, the default is false.
`entityApiName`	string	Required. Indicates the object's API name for enabling topics.

Declarative Metadata Sample Definition

The following is an example of a TopicsForObjects component.

```
<?xml version="1.0" encoding="UTF-8"?>
<TopicsForObjects xmlns="http://soap.sforce.com/2006/04/metadata">
       <enableTopics>false</enableTopics>
       <entityApiName>Account</entityApiName>
</TopicsForObjects>
```

The following is an example `package.xml` that references the previous definition.

```
<?xml version="1.0" encoding="UTF-8"?>
<Package xmlns="http://soap.sforce.com/2006/04/metadata">
<types>
<members>*</members>
<name>TopicsForObjects</name>
</types>
<version>41.0</version>
</Package>
```

Wildcard Support in the Manifest File

This metadata type supports the wildcard character `*` (asterisk) in the `package.xml` manifest file. For information about using the manifest file, see Deploying and Retrieving Metadata with the Zip File.

TransactionSecurityPolicy

Represents a transaction security policy definition. Transaction Security policies give you a way to look through events in your organization and specify actions to take when certain combinations occur.

This type extends the Metadata metadata type and inherits its `fullName` field.

File Suffix and Directory Location

TransactionSecurityPolicy components have the suffix `.transactionSecurityPolicy` and are stored in the `transactionSecurityPolicies` folder.

Version

TransactionSecurityPolicy components are available in API version 35.0 and later.

Fields

Field Name	Field Type	Description
`action`	TransactionSecurityAction	Required. Describes the action to take when the matching Transaction Security policy is triggered.
`active`	boolean	Required. If `true`, the policy is enabled and is actively monitoring its event.
`apexClass`	string	Required for Apex-based policies, and optional for all other policies. The name of the class that implements the `TxnSecurity.PolicyCondition` or `TxnSecurity.EventCondition` interface for this policy. Available in API version 46.0 and later.
`description`	string	A description of the policy.
`developerName`	string	This unique name prevents conflicts with other policies that have the same `masterLabel`. This name can contain only underscores and alphanumeric characters, and must be unique in your org. It must begin with a letter, not include spaces, not end with an underscore, and not contain two consecutive underscores.
`eventName`	TransactionSecurityEventName (enumeration of type string)	Used in Real-Time Event Monitoring only. Indicates the name of the event the policy monitors. This field is available in API 45.0 and later. Valid values are: • `ApiEvent`—Tracks these user-initiated read-only API calls: `query()`, `queryMore()`, and `count()`. Captures API requests through SOAP API, REST API, and Bulk API for the Enterprise and Partner WSDLs. Tooling API calls and API calls originating from a Salesforce mobile app aren't captured.

Field Name	Field Type	Description
		• `ListViewEvent`—Tracks when users access data with list views using Lightning Experience, Salesforce Classic, or the API. It doesn't track list views of Setup entities. • `LoginEvent`—LoginEvent tracks login activity of users who log in to Salesforce. • `ReportEvent`—Tracks when reports are run in your org.
`eventType`	MonitoredEvents (enumeration of type string)	Required for Apex-based policies, and optional for all other policies. Indicates which type of event is being monitored. Valid values are: • `AccessResource`—Notifies you when the selected resource has been accessed. • `AuditTrail`—Reserved for future use. • `DataExport`—Notifies you when the selected object type has been exported using the Data Loader API client. • `Entity`—Notifies you on use of an object type such as an authentication provider or Chatter comment. • `Login`—Notifies you when a user logs in.
`executionUser`	string	Required only for legacy policies (policies that implement the `TxnSecurity.PolicyCondition` interface). The name or ID of an active user who is assigned the Modify All Data and View Setup user permissions. This field is not required or used in enhanced policies (policies that implement the `TxnSecurity.EventCondition` interface).
`flow`	string	Required only for policies of type `CustomConditionBuilderPolicy`. The ID of the Flow object that contains the logic the Condition Builder transaction security policy. Available in API version 46.0 and later.
`masterLabel`	string	The master label for this object. This display value is the internal label that is not translated.
`resourceName`	string	Required for Apex-based policies, and optional for all other policies. A resource used to narrow down the conditions under which the policy triggers. For example, with a `DataExport` event, you can select a resource Lead to specifically monitor export activity occurring on your Lead entities. The resources available depend on the `Event Type` field. Valid resources are grouped below by event type. • AccessResource—ConnectedApplication, Reports • DataExport—Account, Case, Contact, Lead, Opportunity • Entity—AuthProvider, ChatterMessage, FeedComment, FeedItem, Idea, Question • Login—LoginHistory

Field Name	Field Type	Description
type	TxnSecurityPolicyType (enumeration of type string)	The type of validation that the policy uses. The valid values are: • CustomApexPolicy— Created with Apex editor. • CustomConditionBuilderPolicy— Created with Condition Builder. The default value is CustomApexPolicy.

TransactionSecurityAction

Describes the action to take when the matching Transaction Security policy is triggered.

Field Name	Field Type	Description
block	boolean	Required. If true, the requested operation is blocked. This action only applies to Login and AccessResource events.
endSession	boolean	Required. If true, a current session must be closed before a new session can be started. This action only applies to Login events.
freezeUser	boolean	Required. If true, the user that triggered the policy is frozen. This action only applies to Chatter resources for Entity events.
notifications	TransactionSecurityNotification[]	Specifies how to notify the system administrator when the action is triggered. There can be none, one, or multiple notifications.
twoFactorAuthentication	boolean	Required. If true, two-factor authentication is required for a higher level of access before the requested operation can continue. This action only applies to Login and AccessResource events.

TransactionSecurityNotification

Describes who to notify and how to notify them when the matching Transaction Security policy is triggered.

Field Name	Field Type	Description
inApp	boolean	True if an in-app notification is selected.
sendEmail	boolean	True if an email notification is selected.
user	string	The administrator to receive the notification. This user must have the Modify All Data and View Setup permissions.

Declarative Metadata Sample Definition

The following is an example of a TransactionSecurityPolicy component.

```
<?xml version="1.0" encoding="UTF-8"?>
<TransactionSecurityPolicy xmlns="http://soap.sforce.com/2006/04/metadata">
```

```
    <action>
        <block>true</block>
        <endSession>false</endSession>
        <notifications>
            <inApp>false</inApp>
            <sendEmail>true</sendEmail>
            <user>admin@your.org</user>
        </notifications>
        <twoFactorAuthentication>false</twoFactorAuthentication>
    </action>
    <active>true</active>
    <apexClass>TxnSecurityMdApiPolicy</apexClass>
    <eventType>Login</eventType>
    <executionUser>admin@your.org</executionUser>
    <resourceName>LoginHistory</resourceName>
</TransactionSecurityPolicy>
```

The following is an example package manifest used to deploy or retrieve the transaction security metadata for an organization.

```
<?xml version="1.0" encoding="UTF-8"?>
<Package xmlns="http://soap.sforce.com/2006/04/metadata">
    <types>
        <members>MySecurityPolicy</members>
        <name>TransactionSecurityPolicy</name>
    </types>
    <version>35.0</version>
</Package>
```

Wildcard Support in the Manifest File

This metadata type supports the wildcard character `*` (asterisk) in the `package.xml` manifest file. For information about using the manifest file, see Deploying and Retrieving Metadata with the Zip File.

Translations

This metadata type allows you to work with translations for various supported languages. The ability to translate component labels is part of the Translation Workbench. For more information, see “Enable and Disable the Translation Workbench” in the Salesforce online help.

This type extends the Metadata metadata type and inherits its `fullName` field.

Language

A two-character language code identifies each language, such as `en`. A five-character locale code is used for languages that differ depending on the location. For example, en_AU.

Note: Setting a default language is different from setting a default locale. See Select Your Language, Locale, and Currency in Salesforce Help for more information.

Salesforce offers full support for the following languages.

- Chinese (Simplified): `zh_CN`

- Chinese (Traditional): `zh_TW`
- Danish: `da`
- Dutch: `nl_NL`
- English: `en_US`
- Finnish: `fi`
- French: `fr`
- German: `de`
- Italian: `it`
- Japanese: `ja`
- Korean: `ko`
- Norwegian: `no`
- Portuguese (Brazil): `pt_BR`
- Russian: `ru`
- Spanish: `es`
- Spanish (Mexico): `es_MX` Spanish (Mexico) defaults to Spanish for customer-defined translations.
- Swedish: `sv`
- Thai: `th` The Salesforce user interface is fully translated to Thai, but Help is in English.

End-user languages are useful if you have a multilingual organization or partners who speak languages other than your company's default language. For end-user languages, Salesforce provides translated labels for all standard objects and pages, except admin pages, Setup, and Help. When you specify an end-user language, labels and Help that aren't translated appear in English. End-user languages are intended only for personal use by end users. Don't use end-user languages as corporate languages. Salesforce doesn't provide customer support in end-user languages.

End-user languages include:

- Arabic: `ar`
- Bulgarian: `bg`
- Croatian: `hr`
- Czech: `cs`
- English (UK): `en_GB`
- Greek: `el`
- Hebrew: `iw`
- Hungarian: `hu`
- Indonesian: `in`
- Polish: `pl`
- Portuguese (European): `pt_PT`
- Romanian: `ro`
- Slovak: `sk`
- Slovenian: `sl`
- Turkish: `tr`
- Ukrainian: `uk`
- Vietnamese: `vi`

Important: Before enabling end-user languages Arabic and Hebrew, review the right-to-left language support limitations. See Right-to-Left (RTL) Language Support in Salesforce Help for details.

In situations where Salesforce doesn't provide default translations, use platform-only languages to localize apps and custom functionality that you build on the Salesforce Platform. You can translate items such as custom labels, custom objects, and field names. You can also rename most standard objects, labels, and fields. Informational text and non-field label text aren't translatable.

Platform-only languages are available in all places where you can select a language in the application. However, when you select a platform-only language, all standard Salesforce labels default to English or, in select cases, to an end-user or fully supported language.

Note: Language support is closely tied to the API version. For example, we introduced support for Belgian Dutch (nl_BE) in API version 40.0. To take advantage of this language, you must use API version 40.0 or later. In general, we recommend using the most recent version of the API to make the most of our language features.

- Albanian: `sq`
- Afrikaans: `af`
- Arabic (Algeria): `ar_DZ`
- Arabic (Bahrain): `ar_BH`
- Arabic (Egypt): `ar_EG`
- Arabic (Iraq): `ar_IQ`
- Arabic (Jordan): `ar_JO`
- Arabic (Kuwait): `ar_KW`
- Arabic (Lebanon): `ar_LB`
- Arabic (Libya): `ar_LY`
- Arabic (Morocco): `ar_MA`
- Arabic (Oman): `ar_OM`
- Arabic (Qatar): `ar_QA`
- Arabic (Saudi Arabia): `ar_SA`
- Arabic (Sudan): `ar_SD`
- Arabic (Syria): `ar_SY`
- Arabic (Tunisia): `ar_TN`
- Arabic (United Arab Emirates): `ar_AE`
- Arabic (Yemen): `ar_YE`
- Armenian: `hy`
- Basque: `eu`
- Bosnian: `bs`
- Bengali: `bn`
- Burmese: `my`
- Catalan: `ca`
- Chinese (Hong Kong): `zh_HK`
- Chinese (Singapore): `zh_SG`
- Dutch (Belgium): `nl_BE`
- English (Australia): `en_AU`
- English (Canada): `en_CA`

- English (Hong Kong): en_HK
- English (India): en_IN
- English (Ireland): en_IE
- English (Malaysia): en_MY
- English (New Zealand): en_NZ
- English (Philippines): en_PH
- English (Singapore): en_SG
- English (South Africa): en_ZA
- Estonian: et
- Farsi: fa
- French (Belgium): fr_BE
- French (Canada): fr_CA
- French (Luxembourg): fr_LU
- French (Switzerland): fr_CH
- Georgian: ka
- German (Austria): de_AT
- German (Belgium): de_BE
- German (Luxembourg): de_LU
- German (Switzerland): de_CH
- Gujarati: gu
- Hindi: hi
- Icelandic: is
- Irish: ga
- Italian (Switzerland): it_CH
- Kannada: kn
- Latvian: lv
- Lithuanian: lt
- Luxembourgish: lb
- Macedonian: mk
- Malay: ms
- Malayalam: ml
- Maltese: mt
- Marathi: mr
- Montenegrin: sh_ME
- Romanian (Moldova): ro_MD
- Romansh: rm
- Serbian (Cyrillic): sr
- Serbian (Latin): sh
- Spanish (Argentina): es_AR
- Spanish (Bolivia): es_BO

- Spanish (Chile): es_CL
- Spanish (Colombia): es_CO
- Spanish (Costa Rica): es_CR
- Spanish (Dominican Republic): es_DO
- Spanish (Ecuador): es_EC
- Spanish (El Salvador): es_SV
- Spanish (Guatemala): es_GT
- Spanish (Honduras): es_HN
- Spanish (Nicaragua): es_NI
- Spanish (Panama): es_PA
- Spanish (Paraguay): es_PY
- Spanish (Peru): es_PE
- Spanish (Puerto Rico): es_PR
- Spanish (United States): es_US
- Spanish (Uruguay): es_UY
- Spanish (Venezuela): es_VE
- Swahili: sw
- Tagalog: tl
- Tamil: ta
- Te Reo (New Zealand): mi
- Telugu: te
- Urdu: ur
- Welsh: cy
- Xhosa: xh
- Zulu: zu

Important: Before enabling Urdu as a platform-only language, review the right-to-left language support limitations. See Right-to-Left (RTL) Language Support in Salesforce Help for details.

Declarative Metadata File Suffix and Directory Location

Translations are stored in a file with a format of *localeCode*.translation, where *localeCode* is the locale code of the translation language. For example, the file name for German translations is de.translation. The supported locale codes are listed in Language.

Custom object translations are stored in the translations folder in the corresponding package directory.

Version

Translations components are available in API version 14.0 and later.

Fields

Field	Field Type	Description
customApplications	CustomApplicationTranslation[]	A list of custom application translations.
customLabels	CustomLabelTranslation[]	A list of custom label translations.
customPageWebLinks	CustomPageWebLinkTranslation[]	A list of translations for web links defined in a home page component.
customTabs	CustomTabTranslation[]	A list of custom tab translations.
flowDefinitions	FlowDefinitionTranslation[]	A list of flow translations. Only Flow and AutolaunchedFlow types are supported for translation. This field is available in API version 41.0 and later.
fullName	string	Required. The language code; for example, de for German. Inherited from Metadata, this field is not defined in the WSDL for this metadata type. It must be specified when creating, updating, or deleting. See createMetadata() to see an example of this field specified for a call.
globalPicklists	GlobalPicklistTranslation[]	A list of global picklist translations. A global picklist's value set is inherited by all the custom picklist fields that are based on it. This field is available in API version 37.0 only and is removed from later versions.
prompts	PromptTranslation	A list of In-App Guidance prompt translations. This field is available in API version 48.0 and later.
quickActions	GlobalQuickActionTranslation[]	A list of global (rather than object-specific) quick actions.
reportTypes	ReportTypeTranslation[]	A list of report type translations.
scontrols	ScontrolTranslation[]	A list of s-control translations.

CustomApplicationTranslation

CustomApplicationTranslation contains details for a custom application translation. For more details, see CustomApplication.

Field	Field Type	Description
label	string	Required. The translated custom application name. Maximum of 765 characters.
name	string	Required. The name of the custom application.

CustomLabelTranslation

CustomLabelTranslation contains details for a custom label translation. For more details, see CustomLabels.

Field	Field Type	Description
`label`	string	Required. The translated custom label name. Maximum of 765 characters.
`name`	string	Required. The custom label name.

CustomPageWebLinkTranslation

CustomPageWebLinkTranslation contains details for a translation of a web link defined in a home page component. For more details, see CustomPageWebLink.

Field	Field Type	Description
`label`	string	Required. The translated web link.
`name`	string	Required. The name of the web link.

CustomTabTranslation

CustomTabTranslation contains details for a translation of a custom tab. For more details, see CustomTab.

Field	Field Type	Description
`label`	string	Required. The translated custom tab name.
`name`	string	Required. The custom tab name.

FlowDefinitionTranslation

FlowDefinitionTranslation contains details for a translation of a flow definition. For more details, see FlowDefinition.

Available in API version 41.0 and later.

Field	Field Type	Description
`flows`	FlowTranslation[]	A list of flow version translations for the flow definition.
`fullName`	string	Required. The API name for the flow definition.
`label`	string	A translated label for the flow definition. By default, flow definitions inherit the label of the active flow version. If you provide a label here, the definition label no longer inherits changes to the active version label.

FlowTranslation

FlowTranslation contains details for a translation of a flow version. For more details, see Flow.

Available in API version 41.0 and later.

Field	Field Type	Description
`choices`	FlowChoiceTranslation[]	A list of choice translations for the flow version.
`fullName`	string	Required. The API name for the flow version. A unique name for the flow that contains only underscores and alphanumeric characters. The name must be unique across the org, begin with a letter, not include spaces, not end with an underscore, and not contain two consecutive underscores. To deploy or retrieve a version, you can specify the version number. For example, `sampleFlow-3` specifies version 3 of the flow whose unique name is sampleFlow. If you don't specify a version number, the flow is the latest version. In API version 43.0 and earlier, this field included the version number. In API version 44 and later, this field no longer includes the version number.
`label`	string	A translated label for the flow version.
`screens`	FlowScreenTranslation[]	A list of screen translations for the flow version.
`stages`	FlowStageTranslation on page 1100[]	A list of stage translations for the flow version. Available in API version 43.0 and later.

FlowChoiceTranslation

FlowChoiceTranslation contains details for a translation of a choice in a flow version. For more details, see FlowChoice in Flow.

Available in API version 41.0 and later.

Field	Field Type	Description
`choiceText`	string	A translated label for the choice.
`name`	string	Required. A unique name for the choice.
`userInput`	FlowChoiceUserInputTranslation	A translated choice input for the choice.

FlowChoiceUserInputTranslation

FlowChoiceUserInputTranslation contains details for a translation of a choice input. For more details, see FlowChoiceUserInput in Flow.

Available in API version 41.0 and later.

Field	Field Type	Description
`promptText`	string	A translated label for the choice input.
`validationRule`	FlowInputValidationRuleTranslation	A translated validation rule for the choice input.

FlowInputValidationRuleTranslation

FlowInputValidationRuleTranslation contains details for a translation of a validation rule. For more details, see FlowInputValidationRule in Flow.

Available in API version 41.0 and later.

Field	Field Type	Description
`errorMessage`	string	A translated error message for the validation rule.

FlowScreenTranslation

FlowScreenTranslation contains details for a translation of a screen. For more details, see FlowScreen in Flow.

Available in API version 41.0 and later.

Field	Field Type	Description
`fields`	FlowScreenField[]	A list of screen component translations for the screen.
`helpText`	string	Translated help text for the screen.
`name`	string	Required. An API name for the screen.
`pausedText`	string	A translated pause confirmation message for the screen.

FlowScreenFieldTranslation

FlowScreenFieldTranslation contains details for a translation of a screen component. For more details, see FlowScreenField in Flow.

Available in API version 41.0 and later.

Note: Translation isn’t supported for screen components that require Lightning runtime.

Field	Field Type	Description
`fieldText`	string	A translated label for the screen component.
`helpText`	string	Translated help text for the screen component.
`name`	string	Required. An API name for the screen component.
`validationRule`	FlowInputValidationRuleTranslation	Translated validation rule for the screen component.

FlowStageTranslation

FlowStageTranslation contains details for a translation of a stage in a flow version. For more details, see FlowStage in Flow.

Available in API version 43.0 and later.

Field	Field Type	Description
`label`	string	A translated label for the stage.
`name`	string	Required. An API name for the stage.

FlowTextTemplateTranslation

FlowTextTemplateTranslation is available only in flows created via Salesforce Surveys and represents the translation details for the text on all the pages of a survey.

Available in API version 45.0 and later.

Field	Field Type	Description
`name`	string	Required. Unique name for the text template.
`text`	string	Translated text for the text template.

GlobalPicklistTranslation

Note: GlobalPicklistTranslation is available in API version 37.0 only and is removed from later versions.

GlobalPicklistTranslation contains details for a global picklist translation.

Translations are stored in a file with a format of `globalPicklistName__e-lang.objectTranslation`, where `globalPicklistName__e` is the global picklist name, and `lang` is the translation language. To reference a global picklist translation value, use `globalPicklistName__e.value1`, where `value1` is the translated value for the user interface.

Here’s what translations look like for a global picklist.

```
<?xml version="1.0" encoding="UTF-8"?>
<Translations xmlns="http://soap.sforce.com/2006/04/metadata">
<globalPicklists>
    <name>transpicklist</name>
    <picklistValues>
        <masterLabel>Three</masterLabel>
        <translation>Trois</translation>
    </picklistValues>
    <picklistValues>
        <masterLabel>Four</masterLabel>
        <translation>Quatre</translation>
    </picklistValues>
</globalPicklists>
</Translations>
```

Field	Field Type	Description
`name`	string	Required. Represents the name of a global picklist to be translated.
`picklistValues`	PicklistValueTranslation[]	A list of picklist values from global picklists to be translated.

GlobalQuickActionTranslation

GlobalQuickActionTranslation contains details for the translation of a quick action, globally. For more information, see QuickAction.

Field	Field Type	Description
`label`	string	Required. The translated quick action name, globally.
`name`	string	Required. The quick action name.

PromptTranslation

PromptTranslation contains metadata for the translation of a prompt, which is part of In-App Guidance. Available in API Version 48.0 and later.

Field	Field Type	Description
`description`	string	The prompt description
`label`	string	The translated prompt name.
`name`	string	Required. The name of the prompt.
`promptVersions`	PromptVersionTranslation	A list of the prompt version translations.

PromptVersionTranslation

PromptVersionTranslation contains details for translation of a prompt, which is part of In-App Guidance. Available in API Version 48.0 and later.

Field	Field Type	Description
`actionButtonLabel`	string	The label for the prompt's action button.
`actionButtonLink`	string	The URL for the prompt's action button.
`body`	string	The body text of the prompt.
`description`	string	The prompt description.
`dismissButtonLabel`	string	The label for the floating prompt's dismiss button.
`header`	string	The header for the docked prompt.
`label`	string	The translated prompt name.

Field	Field Type	Description
name	string	Required. The name of the prompt.
title	string	The title of the prompt.
videoLink	string	The URL for the docked prompt's video.

ReportTypeTranslation

ReportTypeTranslation contains details for a translation of a custom report type. For more details, see ReportType.

Field	Field Type	Description
description	string	The translated report type description.
label	string	The translated report type name.
name	string	Required. The name of the report type.
sections	ReportTypeSectionTranslation[]	A list of report type section translations.

ReportTypeSectionTranslation

ReportTypeSectionTranslation contains details for a report type section translation.

Field	Field Type	Description
columns	ReportTypeColumnTranslation[]	A list of report type column translations.
label	string	The translated report type section name.
name	string	Required. The name of the report type section.

ReportTypeColumnTranslation

ReportTypeColumnTranslation contains details for a report type column translation.

Field	Field Type	Description
label	string	Required. The translated report type column name.
name	string	Required. The report type column name.

ScontrolTranslation

Important: Visualforce pages supersede s-controls. Organizations that haven't previously used s-controls can't create them. Existing s-controls are unaffected, and can still be edited.

ScontrolTranslation contains details for a translation of an s-control. For more information, see "About S-Controls" in the Salesforce online help.

Field	Field Type	Description
`label`	string	Required. The translated s-control name.
`name`	string	Required. The name of the s-control.

Declarative Metadata Sample Definition

A sample XML definition of a translations component is shown below.

```
<?xml version="1.0" encoding="UTF-8"?>
<Translations xmlns="http://soap.sforce.com/2006/04/metadata">
    <customApplications>
    <label>Angebot-Manager</label>
        <name>Quote Manager</name>
    </customApplications>
    <customLabels>
    <label>Dieses ist ein manuelles Angebot</label>
        <name>quoteManual</name>
    </customLabels>
</Translations>
```

Usage

When you use the `retrieve()` call to get translations in your organization, the files returned in the `.translations` folder only include translations for the other metadata types referenced in `package.xml`. For example, the following `package.xml` file contains `types` elements that match all custom applications, custom labels, web links defined in home page components, custom tabs, report types, and s-controls. Translations for all these metadata types are returned because each metadata type is explicitly listed in `package.xml`.

```
<?xml version="1.0" encoding="UTF-8"?>
<Package xmlns="http://soap.sforce.com/2006/04/metadata">
    <types>
       <members>*</members>
       <name>CustomApplication</name>
    </types>
    <types>
       <members>*</members>
       <name>CustomLabels</name>
    </types>
    <types>
       <members>*</members>
       <name>CustomPageWebLink</name>
    </types>
    <types>
       <members>*</members>
       <name>CustomTab</name>
    </types>
    <types>
       <members>*</members>
       <name>ReportType</name>
    </types>
    <types>
       <members>*</members>
       <name>Scontrol</name>
    </types>
    <types>
       <members>*</members>
        <name>Translations</name>
    </types>
    <version>48.0</version>
</Package>
```

Wildcard Support in the Manifest File

This metadata type supports the wildcard character `*` (asterisk) in the `package.xml` manifest file. For information about using the manifest file, see Deploying and Retrieving Metadata with the Zip File.

SEE ALSO:

CustomLabels

UserCriteria

Represents the member criteria to use in community moderation rules. This type extends the Metadata metadata type and inherits its `fullName` field..

File Suffix and Directory Location

UserCriteria components have the suffix `community_name.user_criteria_developer_name.userCriteria` and are stored in the `UserCriteria` folder.

Version

UserCriteria components are available in API version 39.0 and later.

Special Access Rules

To view, create, edit, and delete moderation rules, you need the Manage Communities or Create and Set Up Communities permission. As of Spring ’20 and later, only users with permission to edit moderation rules can access this object.

Fields

Field Name	Field Type	Description
`creationAgeInSeconds`	int	If specified, includes only users that were created within a specific time frame.
`description`	string	The description of the user criteria.

Field Name	Field Type	Description
lastChatterActivityAgeInSeconds	int	If specified, includes only members that have posted or commented in the community within a specific time frame.
masterLabel	string	Name of the user criteria.
userTypes	NetworkUserType (enumeration of type string)	The member type to use in moderation rules. Valid values are: - Internal - Customer - Partner

Declarative Metadata Sample Definition

The following is an example of a UserCriteria component.

```
<?xml version="1.0" encoding="UTF-8"?>
<UserCriteria xmlns="http://soap.sforce.com/2006/04/metadata">
    <masterLabel>Customer and Partner Members</masterLabel>
    <description>Member criteria matches customer and partner member</description>
    <userTypes>Partner</userTypes>
    <userTypes>Customer</userTypes>
</UserCriteria>
```

WaveApplication

Represents the Analytics application. This type extends the Metadata metadata type and inherits its `fullName` field.

File Suffix and Directory Location

WaveApplication components have the suffix `.wapp` and are stored in the `wave` folder.

Version

WaveApplication components are available in API version 37.0 and later.

Fields

Field Name	Field Type	Description
assetIcon	string	The icon that represents the Analytics application.
description	string	The description that appears in the user interface.
folder	string	The internal api name of the folder or application.
masterLabel	string	The user interface label name of the folder or application.

Field Name	Field Type	Description
`shares`	FolderShare	The folder sharing rules.
`templateOrigin`	string	The internal (unique) name of the template used to create the application. This field is blank if the application was not created from a template.
`templateVersion`	string	The version assigned to the application template by the template's creator. This field is blank if the application was not created from a template.

Declarative Metadata Sample Definition

The following is an example of a WaveApplication component.

```
<?xml version="1.0" encoding="UTF-8"?>
<WaveApplication xmlns="http://soap.sforce.com/2006/04/metadata">
  <assetIcon>/analytics/wave/web/proto/images/app/icons/11.png</assetIcon>
  <description>Application that shows my sales</description>
  <folder>edit</folder>
  <masterLabel>Sales Application</masterLabel>
  <shares>
   <accessLevel>EditAllContents</accessLevel>
   <sharedTo>shareswith@org.ee</sharedTo>
   <sharedToType>User</sharedToType>
  </shares>
</WaveApplication>
```

Wildcard Support in the Manifest File

This metadata type supports the wildcard character `*` (asterisk) in the `package.xml` manifest file. For information about using the manifest file, see Deploying and Retrieving Metadata with the Zip File.

WaveDataflow

Represents the WaveDataflow object in the Analytics application. This type extends the MetadataWithContent metadata type and inherits its `content` and `fullName` fields.

File Suffix and Directory Location

WaveDataflow components have the suffix `.wdf` and are stored in the `wave` folder.

Version

WaveDataflow components are available in API version 37.0 and later.

Fields

Field Name	Field Type	Description
application	string	The name of the Analytics application the dataflow is connected to. This field is available in API version 48.0 and later.
dataflowType	string	The type of the dataflow. Supported types are User and Prepared. The default value is User This field is available in API version 41.0 and later.
description	string	The dataflow description that appears in the user interface.
masterLabel	string	Required. The dataflow name that appears in the user interface.

Declarative Metadata Sample Definition

The following is an example of a WaveDataflow component.

```
<?xml version="1.0" encoding="UTF-8"?>
<WaveDataflow xmlns="http://soap.sforce.com/2006/04/metadata"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">    <content xsi:nil="true"/>
    <description>flow1</description>
    <masterLabel>flow1</masterLabel>
</WaveDataflow>
```

Wildcard Support in the Manifest File

This metadata type supports the wildcard character * (asterisk) in the package.xml manifest file. For information about using the manifest file, see Deploying and Retrieving Metadata with the Zip File.

WaveDashboard

Represents the WaveDashboard object in the Analytics application. This type extends the MetadataWithContent metadata type and inherits its content and fullName fields.

File Suffix and Directory Location

WaveDashboard components have the suffix .wdash and are stored in the wave folder.

Version

WaveDashboard components are available in API version 37.0 and later.

Fields

Field Name	Field Type	Description
`application`	string	Required. The internal name of the application.
`description`	string	The dashboard description that appears in the user interface.
`masterLabel`	string	Required. The dashboard name that appears in the user interface.
`templateAssetSourceName`	string	Links the dashboard to the template used to create it. Null for assets not created from a template.

Declarative Metadata Sample Definition

The following is an example of a WaveDashboard component.

```
<?xml version="1.0" encoding="UTF-8"?>
<WaveDashboard xmlns="http://soap.sforce.com/2006/04/metadata"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
  <content xsi:nil="true"/>
  <application>dev__app</application>
  <masterLabel>Dashboard1</masterLabel>
  <description>somedesc</description>
</WaveDashboard>
```

Wildcard Support in the Manifest File

This metadata type supports the wildcard character `*` (asterisk) in the `package.xml` manifest file. For information about using the manifest file, see Deploying and Retrieving Metadata with the Zip File.

WaveDataset

Represents the WaveDataset object in the Analytics application. This type extends the Metadata metadata type and inherits its `fullName` field.

File Suffix and Directory Location

WaveDataset components have the suffix `.wds` and are stored in the `wave` folder.

Version

WaveDataset components are available in API version 37.0 and later.

Fields

Field Name	Field Type	Description
`application`	string	Required. The internal name of the application.
`description`	string	The dataset description that appears in the user interface.
`masterLabel`	string	Required. The user interface label name of the Dataset.
`templateAssetSourceName`	string	Links the dataset to the template used to create it. Null for assets not created from a template.

Declarative Metadata Sample Definition

The following is an example of a WaveDataset component.

```
<WaveDataset>
    <application>SharedApp</application>
    <description>description</description>
    <masterLabel>dataset1</masterLabel>
</WaveDataset>
```

Wildcard Support in the Manifest File

This metadata type supports the wildcard character `*` (asterisk) in the `package.xml` manifest file. For information about using the manifest file, see Deploying and Retrieving Metadata with the Zip File.

WaveLens

Represents the WaveLens object in the Analytics application. This type extends the MetadataWithContent metadata type and inherits its `content` and `fullName` fields.

File Suffix and Directory Location

WaveLens components have the suffix `.wlens` and are stored in the `wave` folder.

Version

WaveLens components are available in API version 37.0 and later.

Fields

Field Name	Field Type	Description
`application`	string	Required. The internal name of the application.
`datasets`	string	A reference to the dataset used to create this lens.

Field Name	Field Type	Description
`description`	string	The dashboard description that appears in the user interface.
`masterLabel`	string	Required. The user interface label name of the dashboard.
`templateAssetSourceName`	string	Links the lens to the template used to create it. Null for assets not created from a template.
`visualizationType`	string	Required. The visualization type to be used for this lens. Valid values are: • `calheatmap`—Calendar heat map • `comparisontable`—Comparison table • `heatmap`—Heat map • `hbar`—Horizontal bar • `hbarhdot`—Horizontal dot plot • `matrix`—Matrix • `parallelcoords`—Parallel coordinates • `pie`—Donut • `pivottable`—Pivot table • `scatter`—Scatter plot • `stackhbar`—Stacked horizontal bar • `stackvbar`—Stacked vertical bar • `time`—Time line • `valuestable`—Values table • `vbar`—Vertical bar • `vdot`—Vertical dot plot

Declarative Metadata Sample Definition

The following is an example of a WaveLens component.

```
<?xml version="1.0" encoding="UTF-8"?>
<WaveLens xmlns="http://soap.sforce.com/2006/04/metadata"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
  <content xsi:nil="true"/>
  <application>dev__app</application>
  <datasets>dev__abc</datasets>
  <masterLabel>lens1</masterLabel>
  <description>lens in shared app</description>
  <visualizationType>hbar</visualizationType>
</WaveLens>
```

Wildcard Support in the Manifest File

This metadata type supports the wildcard character `*` (asterisk) in the `package.xml` manifest file. For information about using the manifest file, see Deploying and Retrieving Metadata with the Zip File.

WaveRecipe

Represents the WaveRecipe object in the Analytics application. This type extends the MetadataWithContent metadata type and inherits its `content` and `fullName` fields.

File Suffix and Directory Location

WaveRecipe components have the suffix `.wdpr` and are stored in the `wave` folder.

Version

WaveRecipe components are available in API version 41.0 and later.

Fields

Field Name	Field Type	Description
`dataflow`	string	The dataflow ID for the Analytics recipe.
`masterLabel`	string	Required. The recipe name that appears in the user interface.
`securityPredicate`	string	Required. A filter condition that defines row-level access to records in a recipe.
`targetDatasetAlias`	string	The name of the dataset the recipe saves data results into.

Declarative Metadata Sample Definition

The following is an example of a WaveRecipe component.

```
<?xml version="1.0" encoding="UTF-8"?>
<WaveRecipe xmlns="http://soap.sforce.com/2006/04/metadata"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">    <content xsi:nil="true"/>
    <dataflow>02KB0000000b5c7MAA</dataflow>
    <masterLabel>recipe1</masterLabel>
    <securityPredicate>'UserId' == "$User.Id"</securityPredicate>
    <targetDatasetAlias>Dataset One</targetDatasetAlias>
</WaveRecipe>
```

Wildcard Support in the Manifest File

This metadata type supports the wildcard character `*` (asterisk) in the `package.xml` manifest file. For information about using the manifest file, see Deploying and Retrieving Metadata with the Zip File.

WaveTemplateBundle

Represents an Analytics template bundle, which can be used to create Analytics apps. A bundle contains an Analytics template definition and all its related resources.This type extends the Metadata metadata type and inherits its `fullName` field.

File Suffix and Directory Location

An Analytics template bundle is a folder that contains definition files for a template. Unlike other metadata components, a WaveTemplateBundle component isn't represented by a single component file, but instead by a collection of JSON and CSV definition files. Each definition file represents a resource in a template, such as lenses, dashboards, dataflows, and comma-separated values. For example, this directory structure shows the hierarchy of the folders and files for one Analytics Template definition, template1.

```
waveTemplates
    template1
        template-info.json
        variables.json
        ui.json
        extFiles
            PostalCodes.csv
```

Analytics template bundles must be under a top-level folder that's named `waveTemplates`. Each bundle must have its own subfolder under the `waveTemplates` folder and be named with the template's fully qualified API name. The bundle folder must contain a template-info.json file to specify the metadata about the template and the references to other definition files. An entire bundle doesn't have a suffix and definition files can have one of the following suffixes.

Suffix	Component Type
.json	JavaScript Object Notation
.csv	Comma-Separated Values

Version

WaveTemplateBundle components are available in API version 35.0 and later.

Special Access Rules

Definitions can be created in both managed and unmanaged packages.

Fields

Field Name	Field Type	Description
`assetIcon`	string	The icon to use by default for new Analytics apps based on this template. Valid values are `1.png` through `20.png`.
`description`	string	The specification of the template.
`label`	string	Required. The label of the template.

Field Name	Field Type	Description
templateType	string	Required. The type of the template. Valid values are: - App - Dashboard - Lens

Wildcard Support in the Manifest File

This metadata type supports the wildcard character * (asterisk) in the package.xml manifest file. For information about using the manifest file, see Deploying and Retrieving Metadata with the Zip File.

WaveXmd

Represents the WaveXmd object in the Analytics application. This type extends the Metadata metadata type and inherits its fullName field.

File Suffix and Directory Location

WaveXmd components have the suffix .xmd and are stored in the wave folder.

Version

WaveXmd components are available in API version 39.0 and later.

Fields

Field Name	Field Type	Description
application	string	The name of the Analytics application the XMD is associated with. Available in API verison 43.0 and later.
dataset	string	Required. Specifies the dataset associated with this XMD.
datasetConnector	string	The name of the connector source for the dataset.
datasetFullyQualifiedName	string	Specifies the fully qualified name of the dataset version associated with this XMD.
dates	WaveXmdDate	List of dates, with formatting information.
dimensions	WaveXmdDimension	List of dimensions, with formatting information.
measures	WaveXmdMeasure	List of measures, with formatting information.
organizations	WaveXmdOrganization	List of organizations, for multi-organization support.
origin	string	The origin of the dataset version.

Field Name	Field Type	Description
type	string	The XMD type. Valid values are: - System - User - Main - Asset Available in API verison 43.0 and later.
waveVisualization	string	The visualization behavior for Analytics assets. Valid values are: - dashboard - lens Available in API verison 43.0 and later.

WaveXmdDate

WaveXmdDate represents a date.

Field	Field Type	Description
alias	string	Required. Alias of the Date column.
compact	boolean	Whether the date should be displayed in compact form.
dateFieldDay	string	The day field.
dateFieldEpochDay	string	The epoch day field.
dateFieldEpochSecond	string	The epoch second field.
dateFieldFiscalMonth	string	The fiscal month field.
dateFieldFiscalQuarter	string	The fiscal quarter field.
dateFieldFiscalWeek	string	The fiscal week field.
dateFieldFiscalYear	string	The fiscal year field.
dateFieldFullYear	string	The full year field.
dateFieldHour	string	The hour field.
dateFieldMinute	string	The minute field.
dateFieldMonth	string	The month field.
dateFieldQuarter	string	The quarter field.
dateFieldSecond	string	The second field.
dateFieldWeek	string	The week field.
dateFieldYear	string	The year field.

Field	Field Type	Description
description	string	The description of the date column.
firstDayOfWeek	int	Required. Represents the first day of the week.
fiscalMonthOffset	int	Required. Offset number of months for the fiscal year in relation to the calendar year.
isYearEndFiscalYear	boolean	Whether the year end is the fiscal year.
label	string	The label of the date column.
showInExplorer	boolean	Whether or not the date should be shown in the explorer.
sortIndex	int	Required. The index value the system assigns to indicate where the item appears in a list.
type	string	Required. The type of date. Valid values are: • Date—A legacy date type. Available when the time zone is not enabled. • DateOnly—A date type without an associated time. Available when the time zone is enabled. • DateTime—A date type that contains both date and time parts. Available when the time zone is enabled.

WaveXmdDimension

WaveXmdDimension represents a dimension.

Field	Field Type	Description
conditionalFormatting	WaveXmdFormattingProperty	The conditional formatting property for the dimension. . Available in API verison 43.0 and later.
customActions	WaveXmdDimensionCustomAction	Custom actions linked to this dimension.
customActionsEnabled	boolean	Indicates whether the dimension has custom actions enabled.
dateFormat	string	The format used for a date that is a dimension.
description	string	The description of the dimension.
field	string	Required. The field name of the dimension (used in queries).
fullyQualifiedName	string	The fully qualified name of the dimension.
imageTemplate	string	The image template.
isDerived	boolean	Required. Whether this is a derived dimension.
isMultiValue	boolean	Indicates whether the dimension is multi-value.
label	string	The label for the dimension.

Field	Field Type	Description
linkTemplate	string	The template for formatting a link.
linkTemplateEnabled	boolean	Indicates whether the dimension has link templates enabled.
linkTooltip	string	The tooltip to be displayed for links.
members	WaveXmdDimensionMember	The member overrides for a dimension.
origin	string	The origin of this dimension.
recordDisplayFields	WaveXmdRecordDisplayLookup	Ordered list of dimensions. The list defines the default order in which to display the dimensions in the user interface.
recordIdField	string	The record ID for this dimension.
recordOrganizationIdField	string	The record organization ID for this dimension.
salesforceActions	WaveXmdDimensionSalesforceAction	Salesforce actions linked to this dimension.
salesforceActionsEnabled	boolean	Indicates whether the dimension has Salesforce actions enabled.
showDetailsDefaultFieldIndex	int	Default order in which to show the dimensions in the user interface.
showInExplorer	boolean	Indicates whether the dimension is displayed in the explorer.
sortIndex	int	Required. The index value the system assigns to indicate where the item appears in a list.

WaveXmdFormattingProperty

WaveXmdFormattingProperty represents a XMD formatting property for conditional formatting.

Field	Field Type	Description
formattingBins	WaveXmdFormattingBin	The formatting bins for this property.
formattingPredicates	WaveXmdFormattingPredicate	The formatting predicates for this property.
property	string	Required. The property name.
referenceField	string	Required. The reference field for this property.
sortIndex	int	Required. The index value the system assigns to indicate where the item appears in a list.
type	string	Required. The property type.

WaveXmdFormattingBin

WaveXmdFormatttingBin represents a XMD formatting bin for conditional formatting.

Field	Field Type	Description
`bin`	string	Required. The formatting bin.
`formatValue`	string	Required. The format value for the bin.
`label`	string	Required. The label for the bin.
`sortIndex`	int	Required. The index value the system assigns to indicate where the item appears in a list.

WaveXmdFormattingPredicate

WaveXmdFormattingPredicate represents a XMD formatting predicate for conditional formatting.

Field	Field Type	Description
`formatValue`	string	Required. The format value for the predicate.
`operator`	string	Required. The operator for the predicate.
`sortIndex`	int	Required. The index value the system assigns to indicate where the item appears in a list.
`value`	string	Required. The value for the predicate.

WaveXmdDimensionCustomAction

WaveXmdDimensionCustomAction represents a custom action in a dimension.

Field	Field Type	Description
`customActionName`	string	Required. The name of this custom action.
`enabled`	boolean	Required. Indicates whether the action is enabled for a specific dimension.
`icon`	string	The icon for the action.
`method`	string	The method for the action.
`sortIndex`	int	Required. The index value the system assigns to indicate where the item appears in a list.
`target`	string	The target for the action.
`tooltip`	string	The tooltip for the action.
`url`	string	The URL for the action.

WaveXmdDimensionMember

WaveXmdDimensionMember represents a dimension.

Field	Field Type	Description
`color`	string	The color for the member.
`label`	string	The label for the member.
`member`	string	Required. The member value.
`sortIndex`	int	Required. The index value the system assigns to indicate where the item appears in a list.

WaveXmdRecordDisplayLookup

WaveXmdDimensionRecordDisplayLookup represents a record display field.

Field	Field Type	Description
`recordDisplayField`	string	Required. The field to display.
`sortIndex`	int	Required. The index value the system assigns to indicate where the item appears in a list.

WaveXmdDimensionSalesforceAction

WaveXmdDimensionSalesforceAction represents an action in a dimension.

Field	Field Type	Description
`enabled`	boolean	Required. Indicates whether the action is enabled for a specific dimension.
`salesforceActionName`	string	Required. The name of the action.
`sortIndex`	int	Required. The index value the system assigns to indicate where the item appears in a list.

WaveXmdMeasure

WaveXmdMeasure represents a measure.

Field	Field Type	Description
`conditionalFormatting`	WaveXmdFormattingProperty	The conditional formattng for the measure. Available in API verison 43.0 and later.
`dateFormat`	string	The format used for a date that is a measure.
`description`	string	The description of the measure.
`field`	string	Required. The field name of the measure (used in queries).
`formatCustomFormat`	string	The original (XMD 1.1) format array as a single string.

Field	Field Type	Description
formatDecimalDigits	int	The number of digits displayed after the decimal place.
formatDecimalSeparator	string	The custom separator for the decimal place. Available in API verison 48.0 and later.
formatIsNegativeParens	boolean	Indicates whether to display negative numbers with parentheses, rather than a minus sign.
formatPrefix	string	The prefix placed before the field value.
formatSuffix	string	The suffix placed after the field value.
formatThousandsSeparator	string	The custom separator for the thousands place. Available in API verison 48.0 and later.
formatUnit	string	The unit string for the measure. For example, 'cm'.
formatUnitMultiplier	double	The multiplier for the unit.
fullyQualifiedName	string	The fully qualified name of the measure.
isDerived	boolean	Required. Whether this is a derived measure.
label	string	The label for the measure.
origin	string	The origin of the measure.
showDetailsDefaultFieldIndex	int	Default order in which to show the measures in the user interface.
showInExplorer	boolean	Indicates whether the measure is displayed in the explorer.
sortIndex	int	Required. The index value the system assigns to indicate where the item appears in a list.

WaveXmdOrganization

WaveXmdOrganization represents a Salesforce organization.

Field	Field Type	Description
instanceUrl	string	Required. The instance URL for the organization.
label	string	Required. The label for the organization.
organizationIdentifier	string	Required. The ID of the organization.
sortIndex	int	Required. The index value the system assigns to indicate where the item appears in a list.

Declarative Metadata Sample Definition

The following is an example of a WaveXmd component for an Asset XMD belonging to a dashboard.

```
<WaveXmd>
    <dataset xsi:nil="true"/>
    <dimesions>
        <conditionalFormatting>
            <formattingBins>
                <bin>*</bin>
                <formatValue>#FFFFFF</formatValue>
                <label xsi:nil="true"/>
                <sortIndex>0</sortIndex>
            </formattingBins>
            <formattingBins>
                <bin>0</bin>
                <formatValue>#000000</formatValue>
                <label xsi:nil="true"/>
                <sortIndex>1</sortIndex>
            </formattingBins>
            <property>chartColor</property>
            <referenceField>count</referenceField>
            <sortIndex xsi:nil="true"/>
            <type>multiple</type>
        </conditionalFormatting>
        <field>all_1.ALL</field>
        <isDerived>false</isDerived>
        <sortIndex>0</sortIndex>
    </dimensions>
    <measures>
        <conditionalFormatting>
            <formattingBins>
                <bin>*</bin>
                <formatValue>#FFFFFF</formatValue>
                <label xsi:nil="true"/>
                <sortIndex>0</sortIndex>
            </formattingBins>
            <formattingBins>
                <bin>0</bin>
                <formatValue>#000000</formatValue>
                <label xsi:nil="true"/>
                <sortIndex>1</sortIndex>
            </formattingBins>
            <property>chartColor</property>
            <referenceField>count</referenceField>
            <sortIndex xsi:nil="true"/>
            <type>multiple</type>
        </conditionalFormatting>
        <field>all_1.count</field>
        <formatCustomFormat>[&quot;#,###.##%&quot;,1]</formatCustomFormat>
        <isDerived>false</isDerived>
        <sortIndex>0</sortIndex>
    </measures>
    <type>Asset</type>
    <waveVisualization>dashboard</waveVisualization>
</WaveXmd>
```

Wildcard Support in the Manifest File

This metadata type supports the wildcard character `*` (asterisk) in the `package.xml` manifest file. For information about using the manifest file, see Deploying and Retrieving Metadata with the Zip File.

Workflow

Represents the metadata associated with a workflow rule. A workflow rule sets workflow actions into motion when its designated conditions are met. You can configure workflow actions to execute immediately when a record meets the conditions in your workflow rule, or set time triggers that execute the workflow actions on a specific day. Use this metadata type to create, update, or delete workflow rule definitions.

For more information, see "Workflow" in the Salesforce Help. This type extends the Metadata metadata type and inherits its `fullName` field.

When using a manifest file, retrieve all workflow components using the following code:

```
<types>
    <members>*</members>
    <name>Workflow</name>
</types>
```

Declarative Metadata File Suffix and Directory Location

Workflow files have the suffix `.workflow`. There is one file per standard or custom object that has workflow. These files are stored in the `workflows` directory of the corresponding package.

Version

Workflow rules are available in API version 13.0 and later.

Workflow

This metadata type represents the valid types of workflow rules and actions associated with a standard or custom object.

Field Name	Field Type	Description
`alerts`	WorkflowAlert[]	An array of all alerts for the object associated with the workflow.
`fieldUpdates`	WorkflowFieldUpdate[]	An array of all field updates for the object associated with the workflow.
`flowActions`	WorkflowFlowAction[]	An array of flow triggers for the object associated with the workflow. Available in API version 30.0 and later. The pilot program for flow trigger workflow actions is closed. If you've already enabled the pilot in your org, you can continue

Field Name	Field Type	Description
		to create and edit flow trigger workflow actions. If you didn't enable the pilot in your org, use the Flows action in Process Builder instead.
`fullName`	string	The developer name used as a unique identifier for API access. The `fullName` can contain only underscores and alphanumeric characters. It must be unique, begin with a letter, not include spaces, not end with an underscore, and not contain two consecutive underscores. This field is inherited from the Metadata component.
`knowledgePublishes`	WorkflowKnowledgePublish[]	An array of Salesforce Knowledge Workflow Publishes associated with the workflow. Available in API version 27.0 and later.
`outboundMessages`	WorkflowOutboundMessage[]	An array of all of the outbound messages for the object associated with the workflow.
`rules`	WorkflowRule[]	An array of all the objects associated with the workflow.
`tasks`	WorkflowTask[]	An array of all the tasks for the object associated with the workflow.

WorkflowActionReference

WorkflowActionReference represents one of the workflow actions.

Field Name	Field Type	Description
`name`	string	Required. The name of the workflow action.
`type`	WorkflowActionType (enumeration of type string)	Required. Available types of workflow actions: • `Alert` • `FieldUpdate` • `FlowAction`—Available in API version 30.0 and later • `OutboundMessage` • `Task` The pilot program for flow trigger workflow actions is closed. If you've already enabled the pilot in your org, you can continue to create and edit flow trigger workflow actions. If you didn't enable the pilot in your org, use the Flows action in Process Builder instead.

WorkflowAlert

WorkflowAlert represents an email alert associated with a workflow rule.

Field Name	Field Type	Description
`ccEmails`	string[]	Additional email addresses. This field is similar to the CC field in email clients. For the email to be sent successfully, set a value for `ccEmails` or `recipients`. You can set values for both fields. The value of `ccEmails` can include up to five different email addresses.
`description`	string	Required. A description of the email alert. Available in API version 16.0 and later.
`fullName`	string	Required. The developer name used as a unique identifier for API access. The `fullName` can contain only underscores and alphanumeric characters. It must be unique, begin with a letter, not include spaces, not end with an underscore, and not contain two consecutive underscores. This field is inherited from the Metadata component.
`protected`	boolean	Required. Indicates whether this component is protected (`true`) or not (`false`). Protected components cannot be linked to or referenced by components created in the installing organization.
`recipients`	WorkflowEmailRecipient[]	The recipients for the email. For the email to be sent successfully, set a value for `ccEmails` or `recipients`. You can set values for both fields.
`senderAddress`	string	The address in the From field for the email alert. This allows you to use a standard global email address for your organization (such as `support@company.com`) instead of the default From field, which is the email address of the person who updates the record. You can only specify a value in this field if the `senderType` is set to `OrgWideEmailAddress`. See "Organization-Wide Email Addresses" in the Salesforce Help.
`senderType`	ActionEmailSenderType (enumeration of type string)	The email used as the sender's From and Reply-To addresses. The following values are valid: • `CurrentUser`—The email address of the person updating the record. This is the default setting. • `DefaultWorkflowUser`—The email address of the default workflow user. • `OrgWideEmailAddress`—A verified global email address for your organization, such as `support@company.com`.

Field Name	Field Type	Description
`template`	string	Required. Named reference to an EmailTemplate. This email template does not have to exist in the zip file, but it must exist in Metadata API. Lightning email templates aren't packageable. We recommend using a Classic email template.

WorkflowEmailRecipient

WorkflowEmailRecipient represents a recipient for an email alert associated with a workflow rule.

Field Name	Field Type	Description
`field`	string	Name of the field referenced in `type`. The field named should be of the type specified in `type`.
`recipient`	string	The recipients for the email. Depending on the type selected, this may be required.
`type`	ActionEmailRecipientTypes (enumeration of type string)	Named reference to an EmailTemplate component. Valid values are: • `accountOwner` - The email is sent to the record's account owner (for example, the Account owner for an Opportunity). • `accountTeam` - Only applicable on the Account object. The email is sent to everyone on that Account's account team. • `campaignMemberDerivedOwner` - Emails are sent to lead and contact owners when contacts are added to a campaign or in response to a campaign. • `contactLookup` - The email is sent to a contact whose value is looked up from a field on the record. For this value, the `field` field must reference a Contact. • `creator` - The email is sent to the record's creator. • `customerPortalOwner` - The email is sent to a specific self-service portal user. For this value, the recipient field must reference a User (by username), only self-service portal users. • `email` - The email is sent to an email address whose value is looked up from a field on the record. For this value, the `field` field must reference an email field. • `group` - The email is sent to all users in a group. For this value, the recipient field must reference a group (by group name).

Field Name	Field Type	Description
		- opportunityTeam - Only applicable on the Opportunity object. The email is sent to everyone on that Opportunity's opportunity team. - owner - The email is sent to the record's owner. - partnerUser - The email is sent to a specific partner user. For this value, the recipient field must reference a User (by username), only partner users. - portalRole - Like role, but for portal roles only. - portalRoleSubordinates - Like roleSubordinates, but for portal roles only. - role - The email is sent to all users in a role. For this value, the recipient field must reference a Role (in the role hierarchy, by role name). - roleSubordinates - The email is sent to all users in a role subordinates. For this value, the recipient field must reference a Role. - roleSubordinatesInternal - Like roleSubordinates, but for internal portal roles only. - user - The email is sent to a specific user. For this value, the recipient field must reference a User (by username). - userLookup - The email is sent to a user whose value is looked up from a field on the record. For this value, the field field must reference a user foreign key field.

WorkflowFieldUpdate

WorkflowFieldUpdate represents a workflow field update. Field updates allow you to automatically update a field value to one that you specify when a workflow rule is triggered.

Field Name	Field Type	Description
description	string	The description of the field update. This information is useful to track the reasoning for initially configuring the field update.
field	string	Required. The field (on the object for the workflow) to be updated.
formula	string	If the operation field value is Formula, this is set to a formula used to compute the new field value.
fullName	string	Required. The developer name used as a unique identifier for API access. The fullName can contain only underscores and alphanumeric characters. It must be unique, begin with a letter, not include spaces, not end with an underscore, and not contain two consecutive underscores. This field is inherited from the Metadata component.

Field Name	Field Type	Description
literalValue	string	If the operation field value is Literal, this is the literal value for the field.
lookupValue	string	If the operation field value is lookupValue, this is the lookup value that is referenced.
lookupValueType	LookupValueType (enumeration of type string)	The type of object that the lookupValue field value is referencing. The valid values are: • Queue • RecordType • User
name	string	Required. A name for the component. Available in version API 16.0 and later.
notifyAssignee	boolean	Required. Notify the assignee when the field is updated.
operation	FieldUpdateOperation (enumeration of type string)	Required. The operation that computes the value with which to update the field. Valid values are: • Formula - Indicates the field will be set to a formula. If set, the formula must be a valid formula. • Literal - Indicates the field will be set to a literal value. If set, the literalValue must be a valid literal value for this field. • LookupValue - Similar to Literal, but for an object reference, such as a contact, user, account, etc. If set, the lookupValue element must be set. Only User is supported in the current API. • NextValue - Indicates that the field will be set to its next value; this is only allowed when the field update references a picklist. • Null - Indicates the field will be set to null. • PreviousValue - Indicates that the field will be set to its previous value; this is only allowed when the field update references a picklist.
protected	boolean	Required. Indicates whether this component is protected (true) or not (false). Protected components cannot be linked to or referenced by components created in the installing organization.
reevaluateOnChange	boolean	When set to true, if the field update changes the field’s value, all workflow rules on the associated object are re-evaluated. Any workflow rules whose criteria are met as a result of the field value change will be triggered. If any of the triggered workflow rules result in another field update that’s also enabled for workflow rule re-evaluation, a domino effect occurs, and more workflow rules can be re-evaluated as a result of the newly-triggered field update. This cascade of workflow rule

Field Name	Field Type	Description
		re-evaluation and triggering can happen up to five times after the initial field update that started it.
`targetObject`	string	This is set if the change is detected on a child record. If this is set, it points to the foreign key reference on the child object (for example, `EmailMessage.ParentId`) pointing to the parent (for example, Case). When set, the formula is based on the child object (for example, `EmailMessage`). This field is named `sourceField` before version 14.0. The field name change is automatically handled between versions and does not require any manual editing of existing XML component files.

WorkflowFlowAction

Represents a flow trigger, which is a workflow action that launches a flow. Available in API version 30.0 and later. For more information, see the following topics in the Salesforce Help.

- Define a Flow Trigger for Workflow (Pilot)
- Flow Trigger Considerations (Pilot)

Note:

- The pilot program for flow trigger workflow actions is closed. If you've already enabled the pilot in your org, you can continue to create and edit flow trigger workflow actions. If you didn't enable the pilot in your org, use the Flows action in Process Builder instead.
- Test mode for flow triggers isn't supported in the Metadata API. If you want a flow trigger to run the latest flow version when an administrator causes the workflow rule to fire, enable test mode via the user interface after deployment.

Field Name	Field Type	Description
`description`	string	Describes the flow trigger.
`flow`	string	Required. API name of the flow that this workflow action launches.
`flowInputs`	WorkflowFlowActionParameter[]	An array of values to pass into flow variables when launching the flow.
`label`	string	Required. Name of the flow trigger.
`language`	string	Reserved for future use.
`protected`	boolean	Reserved for future use.

WorkflowFlowActionParameter

Represents a value specified in the flow trigger that is passed into a variable when launching the flow.

Note: The pilot program for flow trigger workflow actions is closed. If you've already enabled the pilot in your org, you can continue to create and edit flow trigger workflow actions. If you didn't enable the pilot in your org, use the Flows action in Process Builder instead.

Field Name	Field Type	Description
`name`	string	Required. API name of the flow variable. The flow variable must have `isInput` set to `True`.
`value`	string	Required. Value to assign to the flow variable when launching the flow. If the variable's data type is sObject, `value` must be a merge field that identifies a record—or a lookup relationship field that references a record—of the same object type as the variable. For example: • *`{!this}`*—identifies the record that fired the workflow rule. • *`{!Contact}`*—identifies the contact associated with the record that fired the workflow rule. • *`{!Asset.Account}`*—identifies the account associated with the asset that is associated with the record that fired the workflow rule. • *`{!SomeObject__r}`*—uses a lookup relationship field to identify a custom object record associated with the record that fired the workflow rule. For variables of other data types, you can enter a merge field or a literal value. Manually enter a literal value when the variable should have the same value every time the flow is launched, regardless of which record fired the workflow rule. For example, you can enter *`true`* or *`false`* for a variable of type Boolean. Supported merge fields identify a global variable or a field of the same data type as the flow variable. For example: • *`{!Id}`*—ID of the record that fired the workflow rule. • *`{!Account.Owner.Email}`*—email address of the account owner for the account associated with the record that fired the workflow rule. • *`{!$Organization.Country}`*—country of the organization.

WorkflowKnowledgePublish

WorkflowKnowledgePublish represents Salesforce Knowledge article publishing actions and information. Available in API version 27.0 and later.

Field Name	Field Type	Description
`action`	KnowledgeWorkflowAction (enumeration of type string)	The article publishing actions available when this rule fires. Valid values are: • `PublishAsNew`: Publishes the article as a new article. • `Publish`: Publishes the article as a version of a previously published article.
`description`	string	A brief article description.

Field Name	Field Type	Description
`label`	string	Label that represents the article throughout the Salesforce user interface.
`language`	string	The language of the article.
`protected`	boolean	Required. Indicates whether this component is protected (`true`) or not (`false`). Protected components cannot be linked to or referenced by components created in the installing organization.

WorkflowOutboundMessage

WorkflowOutboundMessage represents an outbound message associated with a workflow rule. Outbound messages are workflow and approval actions that send the information you specify to an endpoint you designate, such as an external service. An outbound message sends the data in the specified fields in the form of a SOAP message to the endpoint. For more information, see "Outbound Message Actions" in the Salesforce Help.

Field Name	Field Type	Description
`apiVersion`	double	Required. The API version of the outbound message. This is automatically set to the current API version when the outbound message is created. Valid API versions for outbound messages are 8.0 and 18.0 or later. This API version is used in API calls back to Salesforce using the enterprise or partner WSDLs. The `API Version` can only be modified by using Metadata API. It can't be modified using the Salesforce user interface. This field is available in API version 18.0 and later. Warning: If you change the `apiVersion` to a version that doesn't support one of the `fields` configured for the outbound message, messages will fail until you update your outbound message listener to consume the updated WSDL. You can monitor the status of outbound messages from Setup by entering *`Outbound Messages`* in the `Quick Find` box, then selecting **Outbound Messages** in Salesforce.
`description`	string	Describes the outbound message.
`endpointUrl`	string	Required. The endpoint URL to which the outbound message is sent.
`fields`	string[]	The named references to the fields that are to be sent.
`fullName`	string	Required. The developer name used as a unique identifier for API access. The `fullName` can contain only underscores and alphanumeric characters. It must be unique, begin with a letter, not include spaces, not end with an underscore, and not contain two consecutive underscores. This field is inherited from the Metadata component.

Field Name	Field Type	Description
`includeSessionId`	boolean	Required. Set if you want the Salesforce *session ID* included in the outbound message. Useful if you intend to make API calls and you do not want to include a username and password.
`integrationUser`	string	Required. The named reference to the user under which this message is sent.
`name`	string	Required. A name for the component. Available in version API 16.0 and later.
`protected`	boolean	Required. Indicates whether this component is protected (`true`) or not (`false`). Protected components cannot be linked to or referenced by components created in the installing organization.
`useDeadLetterQueue`	boolean	This field is only available for organizations with dead letter queue permissions turned on. If set, this outbound message will use the dead letter queue if normal delivery fails.

WorkflowRule

This metadata type represents a workflow rule. This type extends the Metadata metadata type and inherits its `fullName` field.

Field Name	Field Type	Description
`actions`	WorkflowActionReference[]	An array of references for the actions that should happen when this rule fires.
`active`	boolean	Required. Determines if this rule is active.
`booleanFilter`	string	For advanced criteria filter, the boolean formula, for example, `(1 AND 2) OR 3`.
`criteriaItems`	FilterItem[]	An array of the boolean criteria (conditions) under which this rule fires. Note that either this or `formula` must be set.
`description`	string	The description of the workflow rule
`formula`	string	The formula condition under which this rule first (either this or `criteriaItems`) must be set
`fullName`	string	The developer name used as a unique identifier for API access. The `fullName` can contain only underscores and alphanumeric characters. It must be unique, begin with a letter, not include spaces, not end with an underscore, and not contain two consecutive underscores.

Field Name	Field Type	Description
		This field is inherited from the Metadata component.
triggerType	WorkflowTriggerTypes (enumeration of type string)	Under what conditions the trigger fires. Valid values are: • onAllChanges - The workflow rule is considered on all changes. • onCreateOnly - The workflow rule is considered only on create. • onCreateOrTriggeringUpdate - The workflow rule is considered on create and triggering updates.
workflowTimeTriggers	WorkflowTimeTrigger	Represents a set of Workflow actions (Field Updates, Email Alerts, Outbound Messages, and Tasks) that should execute before or after a specified interval of time.

WorkflowTask

This metadata type references an assigned workflow task.

Field Name	Field Type	Description
assignedTo	string	Specifies the user, role, or team to which the workflow rule or action is assigned. The field corresponding to the value specified here must be the same as the specified assignedToType.
assignedToType	ActionTaskAssignedToTypes (enumeration of type string)	Valid string values for this type are: • accountCreator - When set, the task is assigned to the record's account's creator. • accountOwner - When set, the task is assigned to the record's account's owner (Opportunity). • accountTeam - Same as WorkflowAlert type • creator - When set, the task is assigned to the record's creator. • opportunityTeam - Same as WorkflowAlert type • owner - When set, the task is assigned to the record's owner. • partnerUser - When set, the assignedTo field references a User (by username), a partner user. • portalRole - When set, the assignedTo field references a Role (by role name), a portal role.

Field Name	Field Type	Description
		- `role` - When set, the `assignedTo` field references a Role (by role name) - `user` - When set, the `assignedTo` field references a User (by username)
`description`	string	The description of this workflow task.
`dueDateOffset`	int	Required. The offset, in days, from either the trigger date, or the date specified in the (optional) `offsetFromField`. This can be a negative number.
`fullName`	string	Required. The developer name used as a unique identifier for API access. The `fullName` can contain only underscores and alphanumeric characters. It must be unique, begin with a letter, not include spaces, not end with an underscore, and not contain two consecutive underscores. This field is inherited from the Metadata component.
`notifyAssignee`	boolean	Required. Set to send an email notification when the task is assigned.
`offsetFromField`	string	Optional field reference of the date field from which the `dueDate` should be computed.
`priority`	string	Required. The priority to assign the created task.
`protected`	boolean	Required. Indicates whether this component is protected (`true`) or not (`false`). Protected components cannot be linked to or referenced by components created in the installing organization.
`status`	string	Required. The status to assign the created task.
`subject`	string	Required. A subject for the workflow task. It is used if an email notification is sent when the task is assigned. Available in API version 16.0 and later.

WorkflowTimeTrigger

Represents a set of Workflow actions (Field Updates, Email Alerts, Outbound Messages, and Tasks) that should execute before or after a specified interval of time.

Field Name	Field Type	Description
`actions`	WorkflowActionReference[]	An array of references for the actions that should happen when this trigger fires.
`offsetFromField`	string	The date type field name that the time-based workflow triggers off of, i.e. `Created Date`, `Last Modified Date`, `Rule`

Field Name	Field Type	Description
		Trigger Date or a custom date field on the object for which the workflow rule is defined.
timeLength	string	The numeric value of the time after/before the workflow triggers. A negative value represents the time length before the trigger will fire. The timeLength is measured in either hours or days, as specified by workflowTimeTriggerUnit.
workflowTimeTriggerUnit	WorkflowTimeUnits (enumeration of type string)	The unit of time before or after which the time-based workflow will trigger. Valid string values are: • Hours • Days

Declarative Metadata Sample Definition

The following is the definition of a workflow rule:

```
<?xml version="1.0" encoding="UTF-8"?>
<Workflow xmlns="http://soap.sforce.com/2006/04/metadata">
    <alerts>
        <fullName>Another_alert</fullName>
        <description>Another alert</description>
        <protected>false</protected>
        <recipients>
            <type>accountOwner</type>
        </recipients>
        <recipients>
            <field>Contact__c</field>
            <type>contactLookup</type>
        </recipients>
        <recipients>
            <field>Email__c</field>
            <type>email</type>
        </recipients>
        <template>TestEmail/Email Test</template>
    </alerts>
    <fieldUpdates>
        <fullName>Enum_Field_Update</fullName>
        <description>Blah</description>
        <field>EnumField__c</field>
        <name>Enum Field Update</name>
        <notifyAssignee>true</notifyAssignee>
        <operation>NextValue</operation>
        <protected>false</protected>
    </fieldUpdates>
    <fieldUpdates>
        <fullName>Enum_Field_Update2</fullName>
        <description>Blah</description>
        <field>EnumField__c</field>
        <literalValue>PLX2</literalValue>
        <name>Enum Field Update2</name>
        <notifyAssignee>true</notifyAssignee>
        <operation>Literal</operation>
        <protected>false</protected>
    </fieldUpdates>
    <fieldUpdates>
        <fullName>Field_Update</fullName>
        <description>TestField update desc</description>
        <field>Name</field>
        <formula>Name &amp; &quot;Updated&quot;</formula>
        <name>Field Update</name>
        <notifyAssignee>false</notifyAssignee>
        <operation>Formula</operation>
        <protected>false</protected>
    </fieldUpdates>
    <fieldUpdates>
        <fullName>Lookup_On_Contact</fullName>
        <field>RealOwner__c</field>
        <lookupValue>admin@acme.com</lookupValue>
        <name>Lookup On Contact</name>
        <notifyAssignee>false</notifyAssignee>
        <operation>LookupValue</operation>
        <protected>false</protected>
    </fieldUpdates>
    <outboundMessages>
        <fullName>Another_Outbound_message</fullName>
        <description>Another Random outbound.</description>
        <endpointUrl>http://www.test.com</endpointUrl>
        <fields>Email__c</fields>
        <fields>Id</fields>
        <fields>Name</fields>
        <includeSessionId>true</includeSessionId>
        <integrationUser>admin@acme.com</integrationUser>
        <name>Another Outbound message</name>
        <protected>false</protected>
    </outboundMessages>
    <rules>
        <fullName>BooleanFilter</fullName>
        <active>false</active>
        <booleanFilter>1 AND 2 OR 3</booleanFilter>
        <criteriaItems>
            <field>CustomObjectForWorkflow__c.CreatedById</field>
            <operation>notEqual</operation>
        </criteriaItems>
        <criteriaItems>
            <field>CustomObjectForWorkflow__c.CreatedById</field>
            <operation>notEqual</operation>
            <value>abc</value>
        </criteriaItems>
        <criteriaItems>
            <field>CustomObjectForWorkflow__c.CreatedById</field>
            <operation>equals</operation>
            <value>xyz</value>
        </criteriaItems>
        <triggerType>onCreateOrTriggeringUpdate</triggerType>
    </rules>
    <rules>
        <fullName>Custom Rule1</fullName>
        <actions>
            <name>Another_alert</name>
            <type>Alert</type>
        </actions>
        <actions>
            <name>Enum_Field_Update2</name>
            <type>FieldUpdate</type>
        </actions>
        <actions>
            <fullName>Field_Update</name>
                <type>FieldUpdate</type>
        </actions>
        <actions>
            <name>Another_Outbound_message</name>
            <type>OutboundMessage</type>
        </actions>
        <actions>
            <name>Role_task_was_completed</name>
            <type>Task</type>
        </actions>
        <active>true</active>
        <criteriaItems>
            <field>CustomObjectForWorkflow__c.Name</field>
            <operation>startsWith</operation>
            <value>ABC</value>
        </criteriaItems>
        <description>Custom Rule1 desc</description>
        <triggerType>onCreateOrTriggeringUpdate</triggerType>
    </rules>
    <rules>
        <fullName>IsChangedFunctionRule</fullName>
        <active>true</active>
        <description>IsChangedDesc</description>
        <formula>ISCHANGED(Name)</formula>
        <triggerType>onAllChanges</triggerType>
    </rules>
    <tasks>
        <fullName>Another_task_was_completed</fullName>
        <assignedToType>owner</assignedToType>
        <description>Random Comment</description>
        <dueDateOffset>20</dueDateOffset>
        <notifyAssignee>true</notifyAssignee>
        <priority>High</priority>
        <protected>false</protected>
        <status>Completed</status>
        <subject>Another task was completed</subject>
    </tasks>
    <tasks>
        <fullName>Role_task_was_completed</fullName>
        <assignedTo>R11</assignedTo>
        <assignedToType>role</assignedToType>
        <dueDateOffset>-2</dueDateOffset>
        <notifyAssignee>true</notifyAssignee>
        <offsetFromField>CustomObjectForWorkflow__c.CreatedDate</offsetFromField>
        <priority>High</priority>
        <protected>false</protected>
        <status>Completed</status>
        <subject>Role task was completed</subject>
    </tasks>
    <tasks>
        <fullName>User_task_was_completed</fullName>
        <assignedTo>admin@acme.com</assignedTo>
        <assignedToType>user</assignedToType>
        <dueDateOffset>-2</dueDateOffset>
        <notifyAssignee>true</notifyAssignee>
        <offsetFromField>User.CreatedDate</offsetFromField>
        <priority>High</priority>
        <protected>false</protected>
        <status>Completed</status>
        <subject>User task was completed</subject>
    </tasks>
</Workflow>
```

Wildcard Support in the Manifest File

This metadata type supports the wildcard character `*` (asterisk) in the `package.xml` manifest file. For information about using the manifest file, see Deploying and Retrieving Metadata with the Zip File.

WorkSkillRouting

Represents a setup object that stores a set of WorkSkillRoutingAttribute objects. These objects are used to route a work item to an agent who has the skills necessary to take the work. This type extends the Metadata metadata type and inherits its `fullName` field.

File Suffix and Directory Location

WorkSkillRouting components have the suffix `workSkillRouting` and are stored in the `workSkillRoutings` folder.

Version

WorkSkillRouting components are available in API version 46.0 and later.

Fields

Field Name	Field Type	Description
`isActive`	boolean	Required. Indicates whether assignment rules are active and can be evaluated.

Field Name	Field Type	Description
masterLabel	string	Required. Master label for this object. This display value is the internal label that is not translated.
relatedEntity	string	Required. Type of Salesforce object that the attributes are associated with.
workSkillRoutingAttributes	WorkSkillRoutingAttribute[]	A set of mappings between work-item field values and skills. Create one attribute mapping set for each object.

WorkSkillRoutingAttribute

Represents the routing assignments between object attributes and skills. Attributes are used to route a work item to an agent who has the skills necessary to take the work.

Fields

Field Name	Field Type	Description
field	string	Required. Field that this attribute applies to.
isAdditionalSkill	boolean	After a designated timeout period, additional skills are dropped from Omni-Channel routing. The case is then routed to the best-matched agent even if they don't have all the skills.
skill	string	Required. Skill used to route the work item when the attribute maps to the value selected.
skillLevel	int	Level of the skill required. This value can range from 0 to 10.
value	string	Attribute value that is assigned to the selected skill.

Declarative Metadata Sample Definition

The following is an example of a WorkSkillRouting component.

```
<?xml version="1.0" encoding="UTF-8"?>
<WorkSkillRouting xmlns="http://soap.sforce.com/2006/04/metadata">
  <isActive>true</isActive>
  <masterLabel>Attribute setup for skills-based routing for Case object</masterLabel>
  <relatedEntity>Case</relatedEntity>
  <workSkillRoutingAttributes>
    <field>Case.Origin</field>
    <isAdditionalSkill>false</isAdditionalSkill>
    <skill>Technical_Skill</skill>
    <skillLevel>3</skillLevel>
    <value>Web</value>
  </workSkillRoutingAttributes>
</WorkSkillRouting>
```

Wildcard Support in the Manifest File

This metadata type supports the wildcard character `*` (asterisk) in the `package.xml` manifest file. For information about using the manifest file, see Deploying and Retrieving Metadata with the Zip File.

CHAPTER 12 Headers

Use headers in Metadata API calls to set options for each call.

IN THIS SECTION:

`AllOrNoneHeader`

Indicates whether to roll back all metadata changes when some of the records in a call result in failures.

Version

This header is available in API version 34.0 and later.

Supported Calls

createMetadata(), updateMetadata(), upsertMetadata(), deleteMetadata()

Usage

If this header isn't used in API version 34.0 and later, by default a call can save a partial set of records (equivalent to `AllOrNoneHeader=false`)—the records that are processed successfully are saved and records that have failures aren't saved.

Fields

Field Name	Type	Description
`allOrNone`	boolean	Set to `true` to cause all metadata changes to be rolled back if any records in the call cause failures. Set to `false` to enable saving only the records that are processed successfully when other records in the call cause failures.

Sample Code—Java

Add the `AllOrNoneHeader` to the metadata connection before you perform a call as follows:

```
metadataConnection.setAllOrNoneHeader(true);
```

This next example shows how to use the `AllOrNoneHeader` when creating two custom objects. Because the second custom object doesn't have the required `Name` field, the `create()` call can't create this custom object and rolls back the first custom object. The output is shown after this code sample.

```
import com.sforce.soap.metadata.*;
import com.sforce.soap.metadata.Error;
import com.sforce.ws.ConnectionException;

public class CallWithHeader {

    MetadataConnection metadataConnection = null;

    public static void main(String[] args) throws ConnectionException {
     CallWithHeader samples = new CallWithHeader();
        samples.createWithHeader();
    }

    public CallWithHeader() throws ConnectionException {
        metadataConnection = MetadataLoginUtil.login();
    }

    public void createWithHeader() throws ConnectionException {
        // Define two custom objects to be inserted.
        CustomObject co1 = new CustomObject();
        String name1 = "MyCustomObject1";
        co1.setFullName(name1 + "__c");
        co1.setDeploymentStatus(DeploymentStatus.Deployed);
        co1.setDescription("Created by the Metadata API");
        co1.setEnableActivities(true);
        co1.setLabel(name1 + " Object");
        co1.setPluralLabel(co1.getLabel() + "s");
        co1.setSharingModel(SharingModel.ReadWrite);

        CustomField nf = new CustomField();
        nf.setType(FieldType.Text);
        nf.setLabel(co1.getFullName() + " Name");
        co1.setNameField(nf);
        // The second custom object doesn't have a Name field
        CustomObject co2 = new CustomObject();
        String name2 = "MyCustomObject2";
        co2.setFullName(name2 + "__c");
        co2.setDeploymentStatus(DeploymentStatus.Deployed);
        co2.setDescription("Created by the Metadata API");
        co2.setEnableActivities(true);
        co2.setLabel(name2 + " Object");
        co2.setPluralLabel(co2.getLabel() + "s");
        co2.setSharingModel(SharingModel.ReadWrite);


        // Setting the allOrNone header to true to cause
        //  the call to not commit any record if one or more
        //  records in this call have failures.
        metadataConnection.setAllOrNoneHeader(true);

        // Now that the header has been set, make the create call.
        SaveResult[] results = metadataConnection
                .createMetadata(new Metadata[] { co1, co2 });

        // Iterate through the call results
        for (SaveResult r : results) {
            if (r.isSuccess()) {
                System.out.println("Created component: " + r.getFullName());
            } else {
                System.out
                        .println("Errors were encountered while creating "
                                + r.getFullName());
                for (Error e : r.getErrors()) {
                    System.out.println("Error message: " + e.getMessage());
                    System.out.println("Status code: " + e.getStatusCode());
                }
            }
        }
    }

}
```

This is the output that the sample returns. The first record is rolled back and the second has a failure.

```
Errors were encountered while creating MyCustomObject1__c
Error message: Record rolled back because not all records were valid and the request was
using AllOrNone header
Status code: ALL_OR_NONE_OPERATION_ROLLED_BACK
Errors were encountered while creating MyCustomObject2__c
Error message: Must specify a nameField of type Text or AutoNumber
Status code: FIELD_INTEGRITY_EXCEPTION
```

CallOptions

Specifies the API client identifier.

Version

This call is available in all API versions.

Supported Calls

All Metadata API calls.

Fields

Field Name	Type	Description
`client`	string	A value that identifies an API client.

Sample Code—Java

To change the API client ID, add the `CallOptions` header to the metadata connection before you perform a call as follows:

```
metadataConnection.setCallOptions("client ID");
```

DebuggingHeader

Specifies that the deployment result will contain the debug log output, and specifies the level of detail included in the log. The debug log contains the output of Apex tests that are executed as part of a deployment.

Version

This header is available in all API versions.

Supported Calls

deploy()

Fields

Field Name	Type	Description
`categories`	LogInfo[]	A list of log categories with their associated log levels.
`debugLevel`	LogType (enumeration of type string)	Deprecated. This field is provided only for backward compatibility. If you provide values for both `debugLevel` and `categories`, the `categories` value is used. The `debugLevel` field specifies the type of information returned in the debug log. The values are listed from the least amount of

Field Name	Type	Description
		information returned to the most information returned. Valid values include: - None - Debugonly - Db - Profiling - Callout - Detail

LogInfo

Specifies the type and amount of information to be returned in the debug log. The `categories` field takes a list of these objects. LogInfo is a mapping of `category` to `level`.

Element Name	Type	Description
`category`	LogCategory	Specify the type of information returned in the debug log. Valid values are: - Db - Workflow - Validation - Callout - Apex_code - Apex_profiling - Visualforce - System - All
`level`	LogCategoryLevel	Specifies the level of detail returned in the debug log. Valid log levels are (listed from lowest to highest): - NONE - ERROR - WARN - INFO - DEBUG - FINE - FINER - FINEST

Sample Code—Java

Add the DebuggingHeader to the metadata connection before you perform the deploy() call as follows.

```
LogInfo[] logs = new LogInfo[1];
logs[0] = new LogInfo();
logs[0].setCategory(LogCategory.Apex_code);
logs[0].setLevel(LogCategoryLevel.Fine);
metadataConnection.setDebuggingHeader(logs);
```

The result of the deploy() call is obtained by calling checkDeployStatus(). After the deployment finishes, and if tests were run, the response of checkDeployStatus() contains the debug log output in the debugLog field of a DebuggingInfo output header.

SessionHeader

Specifies the session ID that the login call returns. This session ID is used to authenticate all subsequent Metadata API calls.

Version

This header is available in all API versions.

Supported Calls

All Metadata API calls.

Fields

Field Name	Type	Description
sessionId	string	The session ID that the login call returns.

Sample Code—Java

Add the SessionHeader to the metadata connection before you perform a call as follows:

```
metadataConnection.setSessionHeader("<session_ID>");
```

APPENDICES

APPENDIX A CustomObjectTranslation Language Support: Fully Supported Languages

Not every language supports all the possible values for the fields in CustomObjectTranslation. Use this appendix to determine which field values a language supports.

Note: Salesforce offers three levels of language support: fully supported languages, end-user languages, and platform-only languages. This appendix provides information only for fully supported languages. For more information, see Supported Languages

Chinese (Simplified)

`plural`

`false`

Chinese (Traditional)

`plural`

`false`

Danish

`article`

`None`

`Definite`

`Indefinite`

`gender`

`Feminine`

`Neuter`

`plural`

`true`

`false`

Dutch

`gender`

`Feminine`

Neuter

plural

true

false

Finnish

caseType

Ablative

Adessive

Allative

Elative

Essive

Genitive

Illative

Inessive

Nominative

Partitive

Translative

plural

true

false

possessive

None

First

Second

French

gender

Masculine

Feminine

startsWith

Consonant

Vowel

plural

true

false

German

`caseType`

`Accusative`

`Dative`

`Genitive`

`Nominative`

`gender`

`Masculine`

`Feminine`

`Neuter`

`plural`

`true`

`false`

Italian

`gender`

`Masculine`

`Feminine`

`startsWith`

`Consonant`

`Special`

`Vowel`

`plural`

`true`

`false`

Japanese

`plural`

`false`

Korean

`plural`

`false`

Norwegian

article

Definite

Indefinite

None

gender

Masculine

Feminine

Neuter

plural

true

false

Portuguese (Brazil)

gender

Masculine

Feminine

plural

true

false

Russian

caseType

Accusative

Dative

Genitive

Instrumental

Nominative

Prepositional

gender

Masculine

Feminine

Neuter

AnimateMasculine

plural

true

`false`

Spanish

`gender`

`Masculine`

`Feminine`

`plural`

`true`

`false`

Spanish (Mexico)

`gender`

`Masculine`

`Feminine`

`plural`

`true`

`false`

Swedish

`article`

`None`

`Definite`

`Indefinite`

`gender`

`Feminine`

`Neuter`

`plural`

`true`

`false`

Thai

`plural`

`false`

APPENDIX B CustomObjectTranslation Language Support: End-User Languages

Not every language supports all the possible values for the fields in CustomObjectTranslation. Use this appendix to determine which field values a language supports.

Note: Salesforce offers three levels of language support: fully supported languages, end-user languages, and platform-only languages. This appendix provides information only for end-user languages. For more information, see Supported Languages

Arabic

`article`

`Definite`

`None`

`gender`

`Masculine`

`Feminine`

`plural`

`true`

`false`

`possessive`

`None`

`First`

`Second`

Bulgarian

`gender`

`Masculine`

`Feminine`

`Neuter`

`plural`

`true`

`false`

Croatian

`caseType`

`Accusative`

`Dative`

`Genitive`

`Instrumental`

`Locative`

`Nominative`

`gender`

`Feminine`

`Masculine`

`Neuter`

`plural`

`true`

`false`

Czech

`caseType`

`Accusative`

`Dative`

`Genitive`

`Instrumental`

`Locative`

`Nominative`

`gender`

`Masculine`

`Feminine`

`Neuter`

`AnimateMasculine`

`plural`

`true`

`false`

English (UK)

`plural`

`false`

`true`

`startsWith`

`Consonant`

`Vowel`

Greek

`caseType`

`Accusative`

`Genitive`

`Nominative`

`gender`

`Masculine`

`Feminine`

`Neuter`

`plural`

`true`

`false`

Hebrew

`article`

`Definite`

`None`

`gender`

`Masculine`

`Feminine`

`plural`

`true`

`false`

Hungarian

`caseType`

`Ablative`

`Accusative`

`Allative`

`Causalfinal`

`Dative`

`Delative`

`Distributive`

`Elative`

`Essiveformal`

`Illative`

`Inessive`

`Instrumental`

`Nominative`

`Sublative`

`Termanative`

`Translative`

`Superessive`

`plural`

`true`

`false`

`possessive`

`None`

`First`

`Second`

`startsWith`

`Consonant`

`Vowel`

Indonesian

`plural`

`false`

`true`

Polish

`caseType`

`Nominative`

`Accusative`

`Dative`

`Genitive`

`Instrumental`

`Locative`

`gender`

`Masculine`

Feminine

Neuter

AnimateMasculine

plural

true

false

Portuguese (Portugal)

gender

Feminine

Masculine

plural

true

false

Romanian

article

Definite

None

gender

Masculine

Feminine

Neuter

plural

true

false

Slovak

caseType

Accusative

Dative

Genitive

Instrumental

Nominative

Locative

gender

Feminine

Masculine

Neuter

AnimateMasculine

plural

true

false

Slovenian

caseType

Accusative

Dative

Genitive

Instrumental

Nominative

Locative

gender

Feminine

Masculine

Neuter

AnimateMasculine

plural

true

false

Turkish

caseType

Ablative

Accusative

Dative

Genitive

Nominative

Locative

possessive

None

First

Second

plural

`true`

`false`

Ukrainian

caseType

`Accusative`

`Dative`

`Genitive`

`Instrumental`

`Nominative`

`Locative`

gender

`Masculine`

`Feminine`

`Neuter`

`AnimateMasculine`

plural

`true`

`false`

Vietnamese

plural

`false`

APPENDIX C StandardValueSet Names and Standard Picklist Fields

In API version 38.0 and later, standard picklists are represented by the StandardValueSet type. In previous versions, standard picklists are represented by the CustomField type. This table lists the names of standard picklists as standard value sets and their corresponding field names.

Note: The names of standard value sets and picklist fields are case-sensitive.

Standard Value Set Name (API version 38.0 and later)	Field Name (API version 37.0 and earlier)
AccountContactMultiRoles	AccountContactRelation.Roles
AccountContactRole	AccountContactRole.Role
AccountOwnership	Account.Ownership
AccountRating	Account.Rating Lead.Rating
AccountType	Account.Type
AssetStatus	Asset.Status
CampaignMemberStatus	CampaignMember.Status
CampaignStatus	Campaign.Status
CampaignType	Campaign.Type
CaseContactRole	CaseContactRole.Role
CaseOrigin	Case.Origin
CasePriority	Case.Priority
CaseReason	Case.Reason
CaseStatus	Case.Status
CaseType	Case.Type
ContactRole	OpportunityContactRole.Role
ContractContactRole	ContractContactRole.Role
ContractStatus	Contract.Status

Standard Value Set Name (API version 38.0 and later)	Field Name (API version 37.0 and earlier)
EntitlementType	Entitlement.Type
EventSubject	Event.Subject
EventType	Event.Type
FiscalYearPeriodName	Period.PeriodLabel
FiscalYearPeriodPrefix	FiscalYearSettings.PeriodPrefix
FiscalYearQuarterName	Period.QuarterLabel
FiscalYearQuarterPrefix	FiscalYearSettings.QuarterPrefix
IdeaCategory[1]	IdeaTheme.Categories[1]
IdeaMultiCategory	Idea.Categories
IdeaStatus	Idea.Status
IdeaThemeStatus	IdeaTheme.Status
Industry	Account.Industry Lead.Industry
LeadSource	Account.AccountSource Lead.LeadSource Opportunity.Source
LeadStatus	Lead.Status
OpportunityCompetitor	Opportunity.Competitors
OpportunityStage	Opportunity.StageName
OpportunityType	Opportunity.Type
OrderStatus	Not available in 37.0 and earlier
OrderType	Order.Type
PartnerRole	Account.PartnerRole
Product2Family	Product2.Family
QuestionOrigin[1]	Question.Origin[1]
QuickTextCategory	QuickText.Category
QuickTextChannel	QuickText.Channel
QuoteStatus	Quote.Status
RoleInTerritory2	UserTerritory2Association.RoleInTerritory2
ResourceAbsenceType	ResourceAbsence.Type

Standard Value Set Name (API version 38.0 and later)	Field Name (API version 37.0 and earlier)
SalesTeamRole	OpportunityTeamMember.TeamMemberRole UserAccountTeamMember.TeamMemberRole UserTeamMember.TeamMemberRole AccountTeamMember.TeamMemberRole
Salutation	Contact.Salutation Lead.Salutation
ServiceAppointmentStatus	ServiceAppointment.Status
ServiceContractApprovalStatus	ServiceContract.ApprovalStatus
ServTerrMemRoleType	ServiceTerritoryMember.Role
SocialPostClassification	SocialPost.Classification
SocialPostEngagementLevel	SocialPost.EngagementLevel
SocialPostReviewedStatus	SocialPost.ReviewedStatus
SolutionStatus	Solution.Status
TaskPriority	Task.Priority
TaskStatus	Task.Status
TaskSubject	Task.Subject
TaskType	Task.Type
WorkOrderLineItemStatus	WorkOrderLineItem.Status
WorkOrderPriority	WorkOrder.Priority
WorkOrderStatus	WorkOrder.Status
WorkTypeGroupAddInfo	WorkTypeGroup.AdditionalInformation

[1] You can’t read or update this standard value set or picklist field.

GLOSSARY

A |B |C |D |E |F |G |H |I |J |K |L |M |N |O |P |Q |R |S |T |U |V |W |X |Y |Z

A

Ant Migration Tool

A toolkit that allows you to write an Apache Ant build script for migrating Lightning Platform components between a local file system and a Salesforce organization.

Apex

Apex is a strongly typed, object-oriented programming language that allows developers to execute flow and transaction control statements on the Lightning platform server in conjunction with calls to the Lightning Platform API. Using syntax that looks like Java and acts like database stored procedures, Apex enables developers to add business logic to most system events, including button clicks, related record updates, and Visualforce pages. Apex code can be initiated by Web service requests and from triggers on objects.

Apex-Managed Sharing

Enables developers to programmatically manipulate sharing to support their application's behavior. Apex-managed sharing is only available for custom objects.

App

Short for "application." A collection of components such as tabs, reports, dashboards, and Visualforce pages that address a specific business need. Salesforce provides standard apps such as Sales and Service. You can customize the standard apps to match the way you work. In addition, you can package an app and upload it to the AppExchange along with related components such as custom fields, custom tabs, and custom objects. Then, you can make the app available to other Salesforce users from the AppExchange.

AppExchange

The AppExchange is a sharing interface from Salesforce that allows you to browse and share apps and services for the Lightning Platform.

AppExchange Upgrades

Upgrading an app is the process of installing a newer version.

Application Programming Interface (API)

The interface that a computer system, library, or application provides to allow other computer programs to request services from it and exchange data.

Asynchronous Calls

A call that doesn't return results immediately because the operation can take a long time. Calls in the Metadata API and Bulk API are asynchronous.

B

Boolean Operators

You can use Boolean operators in report filters to specify the logical relationship between two values. For example, the AND operator between two values yields search results that include both values. Likewise, the OR operator between two values yields search results that include either value.

Bulk API

The REST-based Bulk API is optimized for processing large sets of data. It allows you to query, insert, update, upsert, or delete a large number of records asynchronously by submitting a number of batches which are processed in the background by Salesforce. See also SOAP API.

C

Class, Apex

A template or blueprint from which Apex objects are created. Classes consist of other classes, user-defined methods, variables, exception types, and static initialization code. In most cases, Apex classes are modeled on their counterparts in Java.

Client App

An app that runs outside the Salesforce user interface and uses only the Lightning Platform API or Bulk API. It typically runs on a desktop or mobile device. These apps treat the platform as a data source, using the development model of whatever tool and platform for which they are designed.

Component, Metadata

A component is an instance of a metadata type in the Metadata API. For example, CustomObject is a metadata type for custom objects, and the `MyCustomObject__c` component is an instance of a custom object. A component is described in an XML file and it can be deployed or retrieved using the Metadata API, or tools built on top of it, such as the Salesforce extensions for Visual Studio Code or the Ant Migration Tool.

Component, Visualforce

Something that can be added to a Visualforce page with a set of tags, for example, `<apex:detail>`. Visualforce includes a number of standard components, or you can create your own custom components.

Component Reference, Visualforce

A description of the standard and custom Visualforce components that are available in your organization. You can access the component library from the development footer of any Visualforce page or the *Visualforce Developer's Guide*.

Controller, Visualforce

An Apex class that provides a Visualforce page with the data and business logic it needs to run. Visualforce pages can use the standard controllers that come by default with every standard or custom object, or they can use custom controllers.

Controlling Field

Any standard or custom picklist or checkbox field whose values control the available values in one or more corresponding dependent fields.

Custom App

See App.

Custom Field

A field that can be added in addition to the standard fields to customize Salesforce for your organization’s needs.

Custom Help

Custom text administrators create to provide users with on-screen information specific to a standard field, custom field, or custom object.

Custom Links

Custom links are URLs defined by administrators to integrate your Salesforce data with external websites and back-office systems. Formerly known as Web links.

Custom Object

Custom records that allow you to store information unique to your organization.

Custom S-Control

Note: S-controls have been superseded by Visualforce pages. After March 2010 organizations that have never created s-controls, as well as new organizations, won't be allowed to create them. Existing s-controls will remain unaffected, and can still be edited.

Custom Web content for use in custom links. Custom s-controls can contain any type of content that you can display in a browser, for example a Java applet, an Active-X control, an Excel file, or a custom HTML Web form.

D

Database

An organized collection of information. The underlying architecture of the Lightning Platform includes a database where your data is stored.

Database Table

A list of information, presented with rows and columns, about the person, thing, or concept you want to track. See also Object.

Data Manipulation Language (DML)

An Apex method or operation that inserts, updates, or deletes records.

Decimal Places

Parameter for number, currency, and percent custom fields that indicates the total number of digits you can enter to the right of a decimal point, for example, 4.98 for an entry of 2. Note that the system rounds the decimal numbers you enter, if necessary. For example, if you enter 4.986 in a field with `Decimal Places` of 2, the number rounds to 4.99. Salesforce uses the round half-up rounding algorithm. Half-way values are always rounded up. For example, 1.45 is rounded to 1.5. –1.45 is rounded to –1.5.

Dependent Field

Any custom picklist or multi-select picklist field that displays available values based on the value selected in its corresponding controlling field.

Developer Edition

A free, fully-functional Salesforce organization designed for developers to extend, integrate, and develop with the Lightning Platform. Developer Edition accounts are available on developer.salesforce.com.

Salesforce Developers

The Salesforce Developers website at developer.salesforce.com provides a full range of resources for platform developers, including sample code, toolkits, an online developer community, and the ability to obtain limited Lightning Platform environments.

Document Library

A place to store documents without attaching them to accounts, contacts, opportunities, or other records.

E

Email Alert

Email alerts are actions that send emails, using a specified email template, to specified recipients.

Email Template

A form email that communicates a standard message, such as a welcome letter to new employees or an acknowledgment that a customer service request has been received. Email templates can be personalized with merge fields, and can be written in text, HTML, or custom format.

Note: Lightning email templates aren’t packageable.

Enterprise Edition

A Salesforce edition designed for larger, more complex businesses.

Enterprise WSDL

A strongly-typed WSDL for customers who want to build an integration with their Salesforce organization only, or for partners who are using tools like Tibco or webMethods to build integrations that require strong typecasting. The downside of the Enterprise WSDL is that it only works with the schema of a single Salesforce organization because it is bound to all of the unique objects and fields that exist in that organization's data model.

Entity Relationship Diagram (ERD)

A data modeling tool that helps you organize your data into entities (or objects, as they are called in the Lightning Platform) and define the relationships between them. ERD diagrams for key Salesforce objects are published in the *SOAP API Developer's Guide*.

Enumeration Field

An enumeration is the WSDL equivalent of a picklist field. The valid values of the field are restricted to a strict set of possible values, all having the same data type.

F

Field

A part of an object that holds a specific piece of information, such as a text or currency value.

Field-Level Security

Settings that determine whether fields are hidden, visible, read only, or editable for users. Available in Professional, Enterprise, Unlimited, Performance, and Developer Editions.

Filter Condition/Criteria

Condition on particular fields that qualifies items to be included in a list view or report, such as "State equals California."

Foreign Key

A field whose value is the same as the primary key of another table. You can think of a foreign key as a copy of a primary key from another table. A relationship is made between two tables by matching the values of the foreign key in one table with the values of the primary key in another.

Formula Field

A type of custom field. Formula fields automatically calculate their values based on the values of merge fields, expressions, or other values.

Function

Built-in formulas that you can customize with input parameters. For example, the DATE function creates a date field type from a given year, month, and day.

G

Gregorian Year

A calendar based on a 12-month structure used throughout much of the world.

H

HTTP Debugger

An application that can be used to identify and inspect SOAP requests that are sent from the AJAX Toolkit. They behave as proxy servers running on your local machine and allow you to inspect and author individual requests.

I

ID

See Salesforce Record ID.

Inline S-Control

> Note: S-controls have been superseded by Visualforce pages. After March 2010 organizations that have never created s-controls, as well as new organizations, won't be allowed to create them. Existing s-controls will remain unaffected, and can still be edited.

An s-control that displays within a record detail page or dashboard, rather than on its own page.

Instance

The cluster of software and hardware represented as a single logical server that hosts an organization's data and runs their applications. The Lightning Platform runs on multiple instances, but data for any single organization is always stored on a single instance.

Integration User

A Salesforce user defined solely for client apps or integrations. Also referred to as the logged-in user in a SOAP API context.

ISO Code

The International Organization for Standardization country code, which represents each country by two letters.

J

Junction Object

A custom object with two master-detail relationships. Using a custom junction object, you can model a “many-to-many” relationship between two objects. For example, you create a custom object called “Bug” that relates to the standard case object such that a bug could be related to multiple cases and a case could also be related to multiple bugs.

K

No Glossary items for this entry.

L

License Management Application (LMA)

A free AppExchange app that allows you to track sales leads and accounts for every user who downloads your managed package (app) from the AppExchange.

License Management Organization (LMO)

The Salesforce organization that you use to track all the Salesforce users who install your package. A license management organization must have the License Management Application (LMA) installed. It automatically receives notification every time your package is installed or uninstalled so that you can easily notify users of upgrades. You can specify any Enterprise, Unlimited, Performance, or Developer Edition organization as your license management organization. For more information, go to http://www.salesforce.com/docs/en/lma/index.htm.

Lightning Platform

The Salesforce platform for building applications in the cloud. Lightning Platform combines a powerful user interface, operating system, and database to allow you to customize and deploy applications in the cloud for your entire enterprise.

List View

A list display of items (for example, accounts or contacts) based on specific criteria. Salesforce provides some predefined views.

In the Agent console, the list view is the top frame that displays a list view of records based on specific criteria. The list views you can select to display in the console are the same list views defined on the tabs of other objects. You cannot create a list view within the console.

Local Project

A `.zip` file containing a project manifest (`package.xml` file) and one or more metadata components.

Locale

The country or geographic region in which the user is located. The setting affects the format of date and number fields, for example, dates in the English (United States) locale display as 06/30/2000 and as 30/06/2000 in the English (United Kingdom) locale.

In Professional, Enterprise, Unlimited, Performance, and Developer Edition organizations, a user's individual `Locale` setting overrides the organization's `Default Locale` setting. In Personal and Group Editions, the organization-level locale field is called `Locale`, not `Default Locale`.

Logged-in User

In a SOAP API context, the username used to log into Salesforce. Client applications run with the permissions and sharing of the logged-in user. Also referred to as an integration user.

Lookup Field

A type of field that contains a linkable value to another record. You can display lookup fields on page layouts where the object has a lookup or master-detail relationship with another object. For example, cases have a lookup relationship with assets that allows users to select an asset using a lookup dialog from the case edit page and click the name of the asset from the case detail page.

M

Managed Package

A collection of application components that is posted as a unit on the AppExchange and associated with a namespace and possibly a License Management Organization. To support upgrades, a package must be managed. An organization can create a single managed package that can be downloaded and installed by many different organizations. Managed packages differ from unmanaged packages by having some locked components, allowing the managed package to be upgraded later. Unmanaged packages do not include locked components and cannot be upgraded. In addition, managed packages obfuscate certain components (like Apex) on subscribing organizations to protect the intellectual property of the developer.

Manifest File

The project manifest file (`package.xml`) lists the XML components to retrieve or deploy when working with the Metadata API, or clients built on top of the Metadata API, such as the Salesforce extensions for Visual Studio Code or the Ant Migration Tool.

Manual Sharing

Record-level access rules that allow record owners to give read and edit permissions to other users who might not have access to the record any other way.

Many-to-Many Relationship

A relationship where each side of the relationship can have many children on the other side. Many-to-many relationships are implemented through the use of junction objects.

Master-Detail Relationship

A relationship between two different types of records that associates the records with each other. For example, accounts have a master-detail relationship with opportunities. This type of relationship affects record deletion, security, and makes the lookup relationship field required on the page layout.

Metadata

Information about the structure, appearance, and functionality of an organization and any of its parts. Lightning Platform uses XML to describe metadata.

Metadata WSDL

A WSDL for users who want to use the Lightning Platform Metadata API calls.

Multitenancy

An application model where all users and apps share a single, common infrastructure and code base.

N

Namespace

In a packaging context, a one- to 15-character alphanumeric identifier that distinguishes your package and its contents from packages of other developers on AppExchange, similar to a domain name. Salesforce automatically prepends your namespace prefix, followed by two underscores ("__"), to all unique component names in your Salesforce organization.

Native App

An app that is built exclusively with setup (metadata) configuration on Lightning Platform. Native apps do not require any external services or infrastructure.

O

Object

An object allows you to store information in your Salesforce organization. The object is the overall definition of the type of information you are storing. For example, the case object allow you to store information regarding customer inquiries. For each object, your organization will have multiple records that store the information about specific instances of that type of data. For example, you might have a case record to store the information about Joe Smith's training inquiry and another case record to store the information about Mary Johnson's configuration issue.

Object-Level Help

Custom help text that you can provide for any custom object. It displays on custom object record home (overview), detail, and edit pages, as well as list views and related lists.

Object-Level Security

Settings that allow an administrator to hide whole objects from users so that they don't know that type of data exists. Object-level security is specified with object permissions.

onClick JavaScript

JavaScript code that executes when a button or link is clicked.

One-to-Many Relationship

A relationship in which a single object is related to many other objects. For example, an account may have one or more related contacts.

Organization-Wide Defaults

Settings that allow you to specify the baseline level of data access that a user has in your organization. For example, you can set organization-wide defaults so that any user can see any record of a particular object that is enabled via their object permissions, but they need extra permissions to edit one.

Outbound Message

An outbound message sends information to a designated endpoint, like an external service. Outbound messages are configured from Setup. You must configure the external endpoint and create a listener for the messages using the SOAP API.

Overlay

An overlay displays additional information when you hover your mouse over certain user interface elements. Depending on the overlay, it will close when you move your mouse away, click outside of the overlay, or click a close button.

Owner

Individual user to which a record (for example, a contact or case) is assigned.

P

Package

A group of Lightning Platform components and applications that are made available to other organizations through the AppExchange. You use packages to bundle an app along with any related components so that you can upload them to AppExchange together.

Partner WSDL

A loosely-typed WSDL for customers, partners, and ISVs who want to build an integration or an AppExchange app that can work across multiple Salesforce organizations. With this WSDL, the developer is responsible for marshaling data in the correct object representation, which typically involves editing the XML. However, the developer is also freed from being dependent on any particular data model or Salesforce organization. Contrast this with the Enterprise WSDL, which is strongly typed.

Picklist

Selection list of options available for specific fields in a Salesforce object, for example, the `Industry` field for accounts. Users can choose a single value from a list of options rather than make an entry directly in the field. See also Master Picklist.

Picklist (Multi-Select)

Selection list of options available for specific fields in a Salesforce object. Multi-select picklists allow users to choose one or more values. Users can choose a value by double clicking on it, or choose additional values from a scrolling list by holding down the CTRL key while clicking a value and using the arrow icon to move them to the selected box.

Picklist Values

Selections displayed in drop-down lists for particular fields. Some values come predefined, and other values can be changed or defined by an administrator.

Primary Key

A relational database concept. Each table in a relational database has a field in which the data value uniquely identifies the record. This field is called the primary key. The relationship is made between two tables by matching the values of the foreign key in one table with the values of the primary key in another.

Production Organization

A Salesforce organization that has live users accessing data.

Professional Edition

A Salesforce edition designed for businesses who need full-featured CRM functionality.

Q

Queue

A holding area for items before they are processed. Salesforce uses queues in a number of different features and technologies.

Query String Parameter

A name-value pair that's included in a URL, typically after a '?' character. For example:

```
https://yourInstance.salesforce.com/001/e?name=value
```

R

Record

A single instance of a Salesforce object. For example, “John Jones” might be the name of a contact record.

Record Name

A standard field on all Salesforce objects. Whenever a record name is displayed in a Lightning Platform application, the value is represented as a link to a detail view of the record. A record name can be either free-form text or an autonumber field. `Record Name` does not have to be a unique value.

Record Type

A record type is a field available for certain records that can include some or all of the standard and custom picklist values for that record. You can associate record types with profiles to make only the included picklist values available to users with that profile.

Record-Level Security

A method of controlling data in which you can allow a particular user to view and edit an object, but then restrict the records that the user is allowed to see.

Recycle Bin

A page that lets you view and restore deleted information. Access the Recycle Bin either by using the link in the sidebar in Salesforce Classic or from the App Launcher in Lightning Experience.

Related Object

Objects chosen by an administrator to display in the Agent console's mini view when records of a particular type are shown in the console's detail view. For example, when a case is in the detail view, an administrator can choose to display an associated account, contact, or asset in the mini view.

Relationship

A connection between two objects, used to create related lists in page layouts and detail levels in reports. Matching values in a specified field in both objects are used to link related data; for example, if one object stores data about companies and another object stores data about people, a relationship allows you to find out which people work at the company.

Relationship Query

In a SOQL context, a query that traverses the relationships between objects to identify and return results. Parent-to-child and child-to-parent syntax differs in SOQL queries.

Report Type

A *report type* defines the set of records and fields available to a report based on the relationships between a primary object and its related objects. Reports display only records that meet the criteria defined in the report type. Salesforce provides a set of pre-defined standard report types; administrators can create custom report types as well.

Role Hierarchy

A record-level security setting that defines different levels of users such that users at higher levels can view and edit information owned by or shared with users beneath them in the role hierarchy, regardless of the organization-wide sharing model settings.

Roll-Up Summary Field

A field type that automatically provides aggregate values from child records in a master-detail relationship.

S

SaaS

See Software as a Service (SaaS).

S-Control

> Note: S-controls have been superseded by Visualforce pages. After March 2010 organizations that have never created s-controls, as well as new organizations, won't be allowed to create them. Existing s-controls will remain unaffected, and can still be edited.

Custom Web content for use in custom links. Custom s-controls can contain any type of content that you can display in a browser, for example a Java applet, an Active-X control, an Excel file, or a custom HTML Web form.

Salesforce Extensions for Visual Studio Code

The Salesforce extension pack for Visual Studio Code includes tools for developing on the Salesforce platform in the lightweight, extensible VS Code editor. These tools provide features for working with development orgs (scratch orgs, sandboxes, and DE orgs), Apex, Aura components, and Visualforce.

Salesforce Record ID

A unique 15- or 18-character alphanumeric string that identifies a single record in Salesforce.

Salesforce SOA (Service-Oriented Architecture)

A powerful capability of Lightning Platform that allows you to make calls to external Web services from within Apex.

Sandbox

A nearly identical copy of a Salesforce production organization for development, testing, and training. The content and size of a sandbox varies depending on the type of sandbox and the edition of the production organization associated with the sandbox.

Search Layout

The organization of fields included in search results, in lookup dialogs, and in the key lists on tab home pages.

Session ID

An authentication token that is returned when a user successfully logs in to Salesforce. The Session ID prevents a user from having to log in again every time they want to perform another action in Salesforce. Different from a record ID or Salesforce ID, which are terms for the unique ID of a Salesforce record.

Session Timeout

The time after login before a user is automatically logged out. Sessions expire automatically after a predetermined length of inactivity, which can be configured in Salesforce from Setup by clicking **Security Controls**. The default is 120 minutes (two hours). The inactivity timer is reset to zero if a user takes an action in the web interface or makes an API call.

Setup

A menu where administrators can customize and define organization settings and Lightning Platform apps. Depending on your organization’s user interface settings, Setup may be a link in the user interface header or in the dropdown list under your name.

Sharing

Allowing other users to view or edit information you own. There are different ways to share data:

- Sharing Model—defines the default organization-wide access levels that users have to each other’s information and whether to use the hierarchies when determining access to data.
- Role Hierarchy—defines different levels of users such that users at higher levels can view and edit information owned by or shared with users beneath them in the role hierarchy, regardless of the organization-wide sharing model settings.

- Sharing Rules—allow an administrator to specify that all information created by users within a given group or role is automatically shared to the members of another group or role.
- Manual Sharing—allows individual users to share records with other users or groups.
- Apex-Managed Sharing—enables developers to programmatically manipulate sharing to support their application's behavior. See Apex-Managed Sharing.

Sharing Model

Behavior defined by your administrator that determines default access by users to different types of records.

Sharing Rule

Type of default sharing created by administrators. Allows users in a specified group or role to have access to all information created by users within a given group or role.

Sites

Salesforce Sites enables you to create public websites and applications that are directly integrated with your Salesforce organization—without requiring users to log in with a username and password.

Snippet

> Note: S-controls have been superseded by Visualforce pages. After March 2010 organizations that have never created s-controls, as well as new organizations, won't be allowed to create them. Existing s-controls will remain unaffected, and can still be edited.

A type of s-control that is designed to be included in other s-controls. Similar to a helper method that is used by other methods in a piece of code, a snippet allows you to maintain a single copy of HTML or JavaScript that you can reuse in multiple s-controls.

SOAP (Simple Object Access Protocol)

A protocol that defines a uniform way of passing XML-encoded data.

Software as a Service (SaaS)

A delivery model where a software application is hosted as a service and provided to customers via the Internet. The SaaS vendor takes responsibility for the daily maintenance, operation, and support of the application and each customer's data. The service alleviates the need for customers to install, configure, and maintain applications with their own hardware, software, and related IT resources. Services can be delivered using the SaaS model to any market segment.

SOQL (Salesforce Object Query Language)

A query language that allows you to construct simple but powerful query strings and to specify the criteria that selects data from the Lightning Platform database.

SOSL (Salesforce Object Search Language)

A query language that allows you to perform text-based searches using the Lightning Platform API.

Standard Object

A built-in object included with the Lightning Platform. You can also build custom objects to store information that is unique to your app.

System Log

Part of the Developer Console, a separate window console that can be used for debugging code snippets. Enter the code you want to test at the bottom of the window and click Execute. The body of the System Log displays system resource information, such as how long a line took to execute or how many database calls were made. If the code did not run to completion, the console also displays debugging information.

T

Test Method

An Apex class method that verifies whether a particular piece of code is working properly. Test methods take no arguments, commit no data to the database, and can be executed by the `runTests()` system method either through the command line or in an Apex IDE, such as the Salesforce extensions for Visual Studio Code.

Translation Workbench

The Translation Workbench lets you specify languages you want to translate, assign translators to languages, create translations for customizations you've made to your Salesforce organization, and override labels and translations from managed packages. Everything from custom picklist values to custom fields can be translated so your global users can use Salesforce in their language.

Trigger

A piece of Apex that executes before or after records of a particular type are inserted, updated, or deleted from the database. Every trigger runs with a set of context variables that provide access to the records that caused the trigger to fire, and all triggers run in bulk mode—that is, they process several records at once, rather than just one record at a time.

Trigger Context Variable

Default variables that provide access to information about the trigger and the records that caused it to fire.

U

Unit Test

A unit is the smallest testable part of an application, usually a method. A unit test operates on that piece of code to make sure it works correctly. See also Test Method.

Unlimited Edition

Unlimited Edition is Salesforce's solution for maximizing your success and extending that success across the entire enterprise through the Lightning Platform.

Unmanaged Package

A package that cannot be upgraded or controlled by its developer.

URL (Uniform Resource Locator)

The global address of a website, document, or other resource on the Internet. For example, http://www.salesforce.com.

URL S-Control

Note: S-controls have been superseded by Visualforce pages. After March 2010 organizations that have never created s-controls, as well as new organizations, won't be allowed to create them. Existing s-controls will remain unaffected, and can still be edited.

An s-control that contains an external URL that hosts the HTML that should be rendered on a page. When saved this way, the HTML is hosted and run by an external website. URL s-controls are also called web controls.

V

Validation Rule

A rule that prevents a record from being saved if it does not meet the standards that are specified.

Visualforce

A simple, tag-based markup language that allows developers to easily define custom pages and components for apps built on the platform. Each tag corresponds to a coarse or fine-grained component, such as a section of a page, a related list, or a field. The components can either be controlled by the same logic that is used in standard Salesforce pages, or developers can associate their own logic with a controller written in Apex.

W

Web Control

See URL S-Control.

Web Links

See Custom Links.

Web Service

A mechanism by which two applications can easily exchange data over the Internet, even if they run on different platforms, are written in different languages, or are geographically remote from each other.

WebService Method

An Apex class method or variable that external systems can use, like a mash-up with a third-party application. Web service methods must be defined in a global class.

Web Services API

A Web services application programming interface that provides access to your Salesforce organization's information. See also SOAP PI and Bulk API.

Web Tab

A custom tab that allows your users to use external websites from within the application.

Workflow Action

A workflow action, such as an email alert, field update, outbound message, or task, fires when the conditions of a workflow rule are met.

Workflow Email Alert

A workflow action that sends an email when a workflow rule is triggered. Unlike workflow tasks, which can only be assigned to application users, workflow alerts can be sent to any user or contact, as long as they have a valid email address.

Workflow Field Update

A workflow action that changes the value of a particular field on a record when a workflow rule is triggered.

Workflow Outbound Message

A workflow action that sends data to an external Web service, such as another cloud computing application. Outbound messages are used primarily with composite apps.

Workflow Queue

A list of workflow actions that are scheduled to fire based on workflow rules that have one or more time-dependent workflow actions.

Workflow Rule

A workflow rule sets workflow actions into motion when its designated conditions are met. You can configure workflow actions to execute immediately when a record meets the conditions in your workflow rule, or set time triggers that execute the workflow actions on a specific day.

Workflow Task

A workflow action that assigns a task to an application user when a workflow rule is triggered.

WSDL (Web Services Description Language) File

An XML file that describes the format of messages you send and receive from a Web service. Your development environment's SOAP client uses the Salesforce Enterprise WSDL or Partner WSDL to communicate with Salesforce using the SOAP API.

X

XML (Extensible Markup Language)

A markup language that enables the sharing and transportation of structured data. All Lightning Platform components that are retrieved or deployed through the Metadata API are represented by XML definitions.

Y

No Glossary items for this entry.

Z

Zip File

A data compression and archive format.

A collection of files retrieved or deployed by the Metadata API. See also Local Project.

INDEX