sforce Web Services Developer's Guide

Version 5.0
Last Update: November 30, 2004

Table of Contents

CHAPTER 1: Getting Started

This topic describes concepts that you need to understand in order to use the sforce Web services API. It contains the following sections:

INTRODUCING THE SFORCE WEB SERVICES API

The sforce Web services API provides programmatic access to your organization's salesforce.com information using a simple, powerful, and secure application programming interface (API).

Customize, Integrate, and Extend Your salesforce.com Solutions

The sforce platform allows you to customize, integrate, and extend your organization's salesforce.com data using the language and platform of your choice.

- **Customize salesforce.com** with custom fields, layouts, and Web integration links to meet specific business requirements.
- **Integrate salesforce.com** with your organization's ERP and finance systems, deliver real-time sales and support information to company portals, and populate critical business systems with customer information.
- **Extend salesforce.com** in presentation, business logic, and data services with new functionality that reflects the business requirements of your organization.

For more information about sforce solutions, developer resources, and community resources, go to http://www.sforce.com. For an introduction to key sforce API concepts, see Chapter 2: Basic Concepts on page 19.

Supported Operations

Using your favorite Web service enabled development environment, you can construct Web service client applications that use standard Web service protocols to programmatically:

- log into the sforce server (login call)
- query your organization's information (query, queryMore, and retrieve calls)
- perform text searches across your organization's information (search call)
- create, update, and delete data (create, update, and delete calls)
- perform various administrative tasks, such as retrieving user information (getUserInfo call), changing passwords (setPassword and resetPassword calls), and getting the server's system time (getServerTimestamp call).
- replicate data locally (getUpdated and getDeleted calls)
- obtain and navigate metadata about your organization's data (describeGlobal and describeSObject calls)

For each operation, client applications submit a synchronous request to the sforce Web service, await the Web service's response, and process the results. The sforce Web service commits any changed data automatically. For detailed information about supported Web service operations, see sforce API Calls on page 22 and sforce Utility API Calls on page 94.

sforce Objects

The sforce Web services API interacts with your organization's data via *objects*, which are programmatic representations of your organization's salesforce.com data. *Object properties* represent fields in those data entities, and client applications set or retrieve data values via these properties. For example, accounts are represented by an Account object, and an Account object has fields that represent the account name, phone number, shipping address, and so on. This document describes how to perform query, insert, update, and delete operations on salesforce.com data via the sforce objects. For detailed information about sforce objects, see sforce Objects on page 101.

Supported salesforce.com Editions

To use the sforce Web services API, your organization must use salesforce.com Enterprise Edition. If you are an existing salesforce.com customer and want to upgrade to Enterprise Edition, contact your account representative.

To develop Web service client applications, it is strongly recommended that you use salesforce.com Developer Edition. Developer Edition provides access to *all* of the features available with Enterprise Edition—it is constrained only by the number of users and the amount of storage space. Developer Edition provides a development context that allows you to build and test your solutions without impacting your organization's live data. Developer Edition accounts are available for free at http://www.sforce.com.

Standards Compliance and Compatible Development Platforms

The sforce Web services API is implemented to comply with SOAP 1.1 (Simple Object Access Protocol), and WSDL 1.1 (Web Service Description Language) specifications. The sforce Web services API works with modern SOAP development environments, including, but not limited to, Visual Studio .NET 2003 and Apache Axis. In this document, we provide Java (Axis) and C# (.NET) examples. To see a complete list of compatible development platforms and more sample code, go to http://www.sforce.com.

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 the sforce Web services API.

If you are using Visual Studio for .NET development, we recommend that you use Visual Studio 2003 or higher.

Go to sforce.com For More Information

The sforce.com Web site provides a full suite of developer toolkits, sample code, community-based support, and other resources to help you with your development projects. Be sure to visit http://www.sforce.com and sign up for a free Developer Edition account.

WHAT'S NEW IN VERSION 5.0

This topic describes what's new in the sforce Web services API version 5.0.

Technical Note on sforce.com

The sforce.com Web site provides a new tech note, entitled: "sforce 5.0: Features and Changes," that describes new and changed features in the sforce Web services API version 5.0. To read this tech note, go to:
http://www.sforce.com/resources/tn-13.jsp

sforce API Support Policy

It is recommended that your client applications use the most recent version of the sforce WSDL file. When a new sforce version is released, you should follow the steps in this section to:

- Regenerate the WSDL file (see Step 2: Generate or Obtain the sforce Web Service WSDL For Your Organization on page 4)
- Import it into your environment (see Step 3: Import the WSDL File Into Your Development Platform on page 4)

Salesforce.com strives to make backwards compatibility easy when using the sforce platform. In the Winter '05 release, programs written with the 4.0 or below version of the API will continue to work unmodified.

Programs that compile and run with "Version N" of the sforce WSDL file can generally be recompiled and run with the sforce "Version N+1" wsdl file without modification. To prevent writing new programs with API functions that have been scheduled for removal (see below), some functions might not appear in the "Version N+1" WSDL file. The "Version N+1" WSDL file will likely contain new functionality not found in "Version N".

In order to mature and improve the sforce service with quality and performance, API functions that are no longer practical or useful may be removed from the sforce service. Functions will generally be supported for a minimum of three years (usually longer) from the date of first release. Advanced notice of API removal will be given at least one year before the API is actually removed. Notification will be given to organizations using these APIs again roughly six months before removal and again roughly three months before removal. Notification will be sent via email to all administrators, posted on the sforce message boards, and published in the sforce newsletter.

QUICK START

This topic tells you what you need to start using the sforce API in your development environment. It includes the following steps:

- Step 1: Obtain a salesforce.com Developer Edition Account
- Step 2: Generate or Obtain the sforce Web Service WSDL For Your Organization
- Step 3: Import the WSDL File Into Your Development Platform
- Step 4: Walk Through the Sample Code

Note

Before you begin building client applications, you need to install your development platform according to its product documentation. In addition, you should read Chapter 2: Basic Concepts on page 19.

Step 1: Obtain a salesforce.com Developer Edition Account

If you are not already a member of the sforce developer community, you need to go to http://www.sforce.com and follow the instructions for signing up for a Developer Edition account. Even if you already have an Enterprise Edition account, it is strongly recommended that you use

Developer Edition for developing, staging, and testing your solutions against sample data to avoid impacting your organization's live data. This is especially true for applications that will be inserting, updating, or deleting data (as opposed to simply reading data).

Step 2: Generate or Obtain the sforce Web Service WSDL For Your Organization

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

WSDL Files for sforce Web Services

There are two sforce Web services for which you can obtain WSDL files for API access:

- **sforce Enterprise Web services API** (Enterprise WSDL)—This API is for most enterprise users who are developing client applications for their organization. The Enterprise WSDL file is a strongly typed representation of your organization's data. It provides information about your schema, data types, and fields to your development environment, allowing for a tighter integration between it and the sforce Web service. As such, this WSDL changes if custom fields or custom objects are added to, renamed, or removed from, your organization's salesforce.com configuration.
- **sforce Partner Web services API** (Partner WSDL)—This API is for salesforce.com partners who are developing client applications for multiple organizations. As a loosely typed representation of the salesforce.com object model, this API can be used to access data within any organization. It is more flexible, although not as easy to use, as its Enterprise counterpart. For more information, see sforce Partner Web Services API on page 186.

Generating the WSDL File for Your Organization

Administrators, and users with the "Modify All Data" permission, can download the Web Services Description Language (WSDL) file to integrate and extend salesforce.com using the sforce API. The WSDL file is dynamically generated based on which type of WSDL file (Enterprise or Partner) you download. The generated WSDL defines all of the API calls, objects (including sforce standard and custom objects), and fields that are available for API access for your organization.

To generate the WSDL file for your organization:

1. Log in to salesforce.com using an account that has system administration access.
2. Click **Setup** | **Integrate | WSDL Generator**.
3. Click the appropriate link to download enterprise WSDL (if you are a salesforce.com customer) or partner WSDL.

   Your browser opens the WSDL file.
4. Save the WSDL file to your local file system.

Note

For the Enterprise WSDL file, if new custom fields or objects are added to, renamed, or removed from your organization's information, you need to regenerate the WSDL file in order to access them.

Step 3: Import the WSDL File Into Your Development Platform

Once you have the WSDL file, you need to import it into your development platform so that your development environment can generate the necessary objects for use in building client Web service applications in that environment. This section provides sample instructions for Apache

Axis and Microsoft Visual Studio. For instructions about other development platforms, see your platform's product documentation.

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

Instructions for Java Environments (Apache Axis)

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

Each SOAP client has its own tool for this process. For Apache Axis, you use the WSDL2Java utility. For more information about using WSDL2Java, see the following URL:

http://ws.apache.org/axis/java/reference.html

Note Before you run WSDL2Java, you must have Axis installed on your system and all of its component JAR files must be referenced in your classpath.

The basic syntax for WSDL2Java is:

```
java –classpath=pathToFirstJAR/FirstJARFilename;pathToSecondJAR/SecondJARFilename
org.apache.axis.wsdl.WSDL2Java pathToWsdl/WsdlFilename
```

For sforce WSDL files, the -a switch is recommended to configure WSDL2Java. The following sample command uses the -a switch:

```
java –classpath=pathToFirstJAR/FirstJARFilename;pathToSecondJAR/SecondJARFilename
org.apache.axis.wsdl.WSDL2Java -a pathToWsdl/WsdlFilename
```

This command will generate a set of folders and Java source code files in the same directory in which it was run. After these files are compiled, they can be included in your Java programs for use in creating client applications.

For most Java development environments, you can use Wizard-based tools for this process instead of the command line. For more information about using WSDL2Java with sforce, visit the message boards at sforce.com.

Instructions for Microsoft Visual Studio

Visual Studio languages access the sforce Web services API through objects that serve as proxies for their server-side counterparts. Before using the sforce Web services API, you must first generate these objects from your organization's WSDL file.

Visual Studio provides two approaches for importing your WSDL file and generating an XML Web service client: an IDE-based approach and a command line approach.

Note Before you begin, the first step is to create a new application or open an existing application in Visual Studio. In addition, you need to have generated the WSDL file, as described in Generating the WSDL File for Your Organization on page 4.

An XML Web service client is any component or application that references and uses an XML Web service. This does not necessarily need to be a client-based application. In fact, in many cases, your XML Web service clients might be other Web applications, such as Web Forms or even other XML Web services. When accessing XML Web services in managed code, a proxy class and the .NET Framework handle all of the infrastructure coding.

To access an XML Web service from managed code:

1. Add a Web reference to your project for the XML Web service that you want to access. The Web reference creates a proxy class with methods that serve as proxies for each exposed method of the XML Web service.
2. Add the namespace for the Web reference.
3. Create an instance of the proxy class and then access the methods of that class as you would the methods of any other class.

To add a Web reference

1. On the Project menu, choose **Add Web Reference.**
2. In the URL box of the Add Web Reference dialog box, type the URL to obtain the service description of the XML Web service you want to access, such as:

   `file:///c:\WSDLFiles\enterprise.wsdl`
3. Click the Go button to retrieve information about the XML Web service.
4. In the Web reference name box, rename the Web reference to `sforce`, which is the namespace you will use for this Web reference.
5. Click **Add Reference** to add a Web reference for the target XML Web service. For more information, see the topic "Adding and Removing Web References" in the Visual Studio documentation.
6. Visual Studio retrieves the service description and generates a proxy class to interface between your application and the XML Web service.

Note

If you are using Visual Basic .Net and the Enterprise WSDL, you will need to modify the generated Web service client to overcome a bug in Visual Studio's client generation utility. The sforce API exposes two objects (Case and Event) whose names conflict with Visual Basic keywords. When the classes that represent these objects are created, Visual Studio wraps the class names with brackets (`[Case]` and `[Event]`). This is the method by which you can re-use keywords.

Unfortunately, in the definition of the `SObject` class, Visual Studio does not wrap these to class references in the `System.Xml.Serialization.XmlIncludeAttribute` that are part of the `SObject` definition. To work around this problem in Visual Studio, you need to edit the `XmlIncludeAttribute` settings for Case and Event as shown below. This does note apply to C# and only applies when using the Enterprise version of the sforce WSDL.

```
System.Xml.Serialization.XmlIncludeAttribute(GetType([Event])), _
System.Xml.Serialization.XmlIncludeAttribute(GetType([Case])), _
```

Step 4: Walk Through the Sample Code

Once you have imported your WSDL file, you can begin building client applications that use the sforce Web services API. The fastest way is to learn by example—start by walking through the code example described in Sample Code Walkthrough on page 6.

SAMPLE CODE WALKTHROUGH

This topic walks through the following code samples that use the sforce API:

- Java Sample Code (Apache Axis)
- C# Sample Code

Java Sample Code (Apache Axis)

This section walks through a sample Java client application that uses the Apache Axis SOAP client. The purpose of this sample application is to show the required steps for logging into the sforce server and to demonstrate the invocation and subsequent handling of several sforce API calls. This sample application performs the following main tasks:

1. Prompts the user for their salesforce.com user name and password.
2. Calls login to log in to the sforce single login server and, if the login succeeds:
   - Sets the returned `sessionID` into the session header, which is required for session authentication on subsequent API calls.
   - Resets the sforce Web service endpoint to the returned `serverUrl`, which is the server that will be the target of subsequent API calls.

All client applications that access the sforce Web services API *must* complete the tasks in this step before attempting any subsequent API calls.

3. Calls describeGlobal to retrieve a list of all available objects for the organization's data.
4. Calls describeSObject to retrieve metadata (field list and object properties) for a specified sforce object.
5. Calls query, passing a simple query string (`"select FirstName, LastName from Contact"`), and iterating through the returned QueryResult.

In the following sample code, sforce API calls and other significant code is identified in a **bold** font. In addition, note the error handling code that follows each API call.

The sample client application begins by importing the necessary packages and objects.

```
package com.doc.samples;

import java.io.BufferedReader;
import java.io.IOException;
import java.io.InputStreamReader;
import java.net.MalformedURLException;
import java.net.URL;
import java.rmi.RemoteException;
import java.util.Date;

import javax.xml.rpc.ServiceException;

import org.w3c.dom.Element;

import com.sforce.soap.enterprise.DescribeGlobalResult;
import com.sforce.soap.enterprise.DescribeSObjectResult;
import com.sforce.soap.enterprise.Field;
import com.sforce.soap.enterprise.FieldType;
import com.sforce.soap.enterprise.GetUserInfoResult;
import com.sforce.soap.enterprise.ID;
import com.sforce.soap.enterprise.LoginResult;
import com.sforce.soap.enterprise.PicklistEntry;
import com.sforce.soap.enterprise.QueryResult;
import com.sforce.soap.enterprise.ResetPasswordResult;
import com.sforce.soap.enterprise.SaveResult;
import com.sforce.soap.enterprise.SetPasswordResult;
import com.sforce.soap.enterprise.SforceServiceLocator;
import com.sforce.soap.enterprise.SoapBindingStub;
import com.sforce.soap.enterprise._QueryOptions;
import com.sforce.soap.enterprise._SessionHeader;
import com.sforce.soap.enterprise.fault.ExceptionCode;
import com.sforce.soap.enterprise.fault.LoginFault;
import com.sforce.soap.enterprise.sobject.Account;
import com.sforce.soap.enterprise.sobject.Contact;
import com.sforce.soap.enterprise.sobject.SObject;
import com.sforce.soap.enterprise.sobject.Task;
import com.sforce.soap.enterprise.sobject.User;

/**
 * <p>Title: sforce Login Sample</p>
 * <p>Description: Console application illustrating login, session management and
server redirection.</p>
 * <p>Copyright: Copyright (c) 2003</p>
 * <p>Company: salesforce.com</p>
 * @author Dave Carroll
 * @version 5.0
 */
```



```
public class Samples {

  private SoapBindingStub binding;
  private LoginResult loginResult = null;
  private String userName = "";
  private String password = "";
  private boolean loggedIn = false;
  private GetUserInfoResult userInfo = null;
  private ID[] accounts = null;
  private ID[] contacts = null;
  private ID[] tasks = null;

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

  public Samples() {
  }

  public static void main(String[] args) {
    Samples samples1 = new Samples();
    samples1.run();
  }
```

The sample client application retrieves the user's login credentials.

```
  // Helper function for retrieving user input from the console
  String getUserInput(String prompt) {
    print(prompt, false);
    try {
      return rdr.readLine();
    }
    catch (IOException ex) {
      return null;
    }
  }

  /** The login function is used to obtain a token from salesforce.com.
   *  This token must be passed to all other calls to provide
   *  authentication and is valid for 2 hours.
   */
  private boolean login() {

    userName = getUserInput("Enter user name: ");
    password = getUserInput("Enter password: ");
```

Next, the sample client application initializes the binding stub. This is our main interface to the Web service through which all calls are made. The `getSoap` method takes an optional parameter (a `java.net.URL`), which is the endpoint of the Web service. For the login call, it always starts with http(s)//www.salesforce.com. After logging in, the sample client application changes the endpoint to the one specified in the returned LoginResult object.

```
          binding = (SoapBindingStub)
          new SforceServiceLocator().getSoap();

    // Time out after a minute
    binding.setTimeout(60000);

    // Test operation
    try {
```



```
      print("LOGGING IN NOW....", true);
      loginResult = binding.login(userName, password);
    }
    catch (LoginFault ex) {
        //The LoginFault derives from AxisFault
        ExceptionCode exCode = ex.getExceptionCode();
        if (exCode == ExceptionCode.FUNCTIONALITY_NOT_ENABLED ||
          exCode==ExceptionCode.INVALID_CLIENT ||
          exCode==ExceptionCode.INVALID_LOGIN ||
          exCode==ExceptionCode.LOGIN_DURING_RESTRICTED_DOMAIN ||
          exCode==ExceptionCode.LOGIN_DURING_RESTRICTED_TIME ||
          exCode==ExceptionCode.ORG_LOCKED ||
          exCode==ExceptionCode.PASSWORD_LOCKOUT ||
          exCode==ExceptionCode.SERVER_UNAVAILABLE ||
          exCode==ExceptionCode.TRIAL_EXPIRED ||
          exCode==ExceptionCode.UNSUPPORTED_CLIENT){
          System.out.println("Please make sure that you have a valid user id and
password.");
        } else {
             //Write the fault code to the console
             System.out.println(ex.getExceptionCode());
             //Write the fault message to the console
             System.out.println(ex.getMessage());
          System.out.println("An unexpected error has occurred." +
ex.getMessage());
        }
          return false;
    } catch (RemoteException ex) {
    System.out.println("An unexpected error has occurred." + ex.getMessage());
        return false;
    }
```

Once the client application has logged in successfully, it will use the results of the login call to reset the endpoint of the service to the virtual server instance that is servicing our organization's data. To do this, the client application sets the ENDPOINT_ADDRESS_PROPERTY of the binding object using the URL returned from the LoginResult.

```
binding._setProperty(soapBindingStub.ENDPOINT_ADDRESS_PROPERTY,
loginResult.getServerUrl());
```

The sample client application now has an instance of the `SoapBindingStub` that is pointing to the correct endpoint. Next, the sample client application sets a persistent Soap header (to be included on all subsequent calls that are made with the `SoapBindingStub`) that contains the valid sessionId for our login credentials. To do this, the sample client application creates a new `_SessionHeader` object and set its `sessionId` property to the `sessionId` property from the LoginResult object. Next, the sample client application calls the `setHeader` method of the `SoapBindingStub` to add the header to all subsequent method calls. This header will persist until the `SoapBindingStub` is destroyed or until the header is explicitly removed. The `"SessionHeader"` parameter is the name of the header to be added.

```
    //Create a new session header object and add the session id
    //from the login reutrn object
    _SessionHeader sh = new _SessionHeader();
    sh.setSessionId(loginResult.getSessionId());

    //set the session header for subsequent call authentication
    binding.setHeader(new
SforceServiceLocator().getServiceName().getNamespaceURI(), "SessionHeader", sh);

    //return true to indicate that we are logged in, pointed
    //at the right url and have our security token in place.
```



```
    return true;
  }
```

To determine the objects that are available to the logged in user, the sample client application executes a describeGlobal call, which returns all of the objects that are visible to the logged in user (see Factors that Affect Data Access on page 23). This call should not be made more than once per session, as the data returned from the call likely does not change frequently. The DescribeGlobalResult is simply echoed to the console.

```
  private void describeGlobalSample() {
    try
    {
      DescribeGlobalResult describeGlobalResult = null;
    describeGlobalResult = binding.describeGlobal();

      String[] types = describeGlobalResult.getTypes();

      for (int i=0;i<types.length;i++)
          System.out.println(types[i]);

      System.out.println("\nDescribe global was successful.\n\nHit the enter key
to conutinue....");
      getUserInput();
    }
    catch (Exception ex)
    {
      System.out.println("\nFailed to return types, error message was: \n" +
ex.getMessage());
      System.out.println("\nHit return to continue...");
      getUserInput();
    }
  }
```

The following code segment illustrates the type of metadata information that can be obtained for each object available to the user. The sample client application executes a describeSObject call on a given object and then echoes the returned metadata information to the console. Object metadata information includes permissions, field types and lengths, and available values for picklist fields and types for `referenceTo` fields.

```
  private void describeSample() {

    String objectToDescribe = getUserInput("\nType the name of the object to
describe (try Account): ");
    try {
      DescribeSObjectResult descSObjectRslt = null;
    descSObjectRslt= binding.describeSObject(objectToDescribe);

      if (!(descSObjectRslt == null)) {

      //Report object level information
        Field[] fields=descSObjectRslt.getFields();
        String objectName=descSObjectRslt.getName();
        boolean isActivateable=descSObjectRslt.isActivateable();
        boolean isCreateable=descSObjectRslt.isCreateable();
        boolean isCustom=descSObjectRslt.isCustom();
        boolean isDeleteable=descSObjectRslt.isDeletable();
        boolean isQueryable=descSObjectRslt.isQueryable();
        boolean isReplicateable=descSObjectRslt.isReplicateable();
        boolean isRetrieveable = descSObjectRslt.isRetrieveable();
        boolean isSearchable = descSObjectRslt.isSearchable();
```



```
            boolean isUndeleteable = descSObjectRslt.isUndeletable();
            boolean isUpdateable = descSObjectRslt.isUpdateable();

            System.out.println("Metadata for " + objectToDescribe + " object:\n");
            System.out.println("Object name = " + objectName);
            System.out.println("Number of fields = " + fields.length);
            System.out.println("Object can be activated = " + isActivateable);
            System.out.println("Can create rows of data = " + isCreateable);
            System.out.println("Object is custom object = " + isCustom);
            System.out.println("Can delete rows of data = " + isDeleteable);
            System.out.println("Can query for rows of data = " + isQueryable);
            System.out.println("Object used in replication = " + isReplicateable);
            System.out.println("Can retrieve object = " + isRetrieveable);
            System.out.println("Can search object = " + isSearchable);
            System.out.println("Can un-delete = " + isUndeleteable);
            System.out.println("Can update = " + isUpdateable);
            System.out.println("\nField metadata for " + objectToDescribe + "
object:\n");

          //Report information about each field
            if (!(fields == null)) {
              for (int i=0;i<fields.length;i++) {
                Field field = fields[i];
                int byteLength = field.getByteLength();
                int digits = field.getDigits();
                String label = field.getLabel();
                int length = field.getLength();
                String name = field.getName();
                PicklistEntry[] picklistValues = field.getPicklistValues();
                int precision = field.getPrecision();
                String[] referenceTos = field.getReferenceTo();
                int scale = field.getScale();
                FieldType fieldType = field.getType();
                boolean IsCreateable = field.isCreateable();
                boolean IsCustom = field.isCustom();
                boolean IsFilterable = field.isFilterable();
                boolean IsNillable = field.isNillable();
                boolean IsRequired = field.isRequired();
                boolean IsRestrictedPicklist=field.isRestrictedPicklist();
                boolean IsSelectable = field.isSelectable();
                boolean IsUpdateable = field.isUpdateable();

            System.out.println("*************  New Field  ***************", true);
                System.out.println("Name            = " + name);
                System.out.println("Label           = " + label);
                System.out.println("Length          = " + length);
                System.out.println("Bytelength      = " + byteLength);
                System.out.println("Digits          = " + digits);
                System.out.println("Precision       = " + precision);
                System.out.println("Scale           = " + scale);
                System.out.println("Field type      = " + fieldType);

                if (picklistValues != null) {
                  System.out.println("Picklist values = ");
                  for (int j=0;j<picklistValues.length;j++) {
                    if (picklistValues[j].getLabel() != null)
                      prSystem.out.println("    Item:  " +
picklistValues[j].getLabel(), true);
                    else
                     System.out.println("    Item:  " + picklistValues[j].getValue());
```



```
                System.out.println("        value           = " +
picklistValues[j].getValue());
                System.out.println("        is default      = " +
picklistValues[j].isDefaultValue());
              }
            }
            if (referenceTos != null) {
              System.out.println("Field references the following objects:");
              for (int j=0;j<referenceTos.length;j++)
                System.out.println("    " + referenceTos[j]);
            }
            System.out.println("\n");
          }
          System.out.println("\nDescribe " + objectToDescribe + " was
successful.\n\nHit the enter key to conutinue....");
          getUserInput();
        }
      }
    } catch (Exception ex) {
      System.out.println("\nFailed to " + objectToDescribe + " description, error
message was: \n" + ex.getMessage);
      System.out.println("\nHit return to continue...");
      getUserInput();
    }
  }
```

The sample client application executes a query by invoking the query call, passing a simple query string (**`"select FirstName, LastName from Contact"`**), and iterating through the returned QueryResult.

```
  private void querySample() {

    QueryResult qr = null;
    _QueryOptions qo = new _QueryOptions();
    qo.setBatchSize(new Integer(3));
    binding.setHeader(new
SforceServiceLocator().getServiceName().getNamespaceURI(), "QueryOptions", qo);

    try {
      qr = binding.query("select FirstName, LastName from Contact");
    boolean done = false;
      if (qr.getSize() > 0){
        System.out.println("Logged in user can see " + qr.getRecords().length + "
contact records.");
      while (!done) {
          for (int i=0;i<qr.getRecords().length;i++) {
            Contact con = (Contact)qr.getRecords(i);
            String fName = con.getFirstName();
            String lName = con.getLastName();
            if (fName == null)
              print("Contact " + (i + 1) + ": " + lName);
            else
              print("Contact " + (i + 1) + ": " + fName + " " + lName);
          }
        if (qr.isDone()) {
        done = true;
        } else {
        qr = binding.queryMore(qr.getQueryLocator());
        }
        }
```

```
    }
      else {
      System.out.println("No records found.");
      }
      System.out.println("\nQuery succesfully executed.");
      System.out.println("\nHit return to continue...");
      getUserInput();
    }
    catch (RemoteException ex) {
      System.out.println("\nFailed to execute query succesfully, error message
was: \n" + ex.getMessage());
      System.out.println("\nHit return to continue...");
      getUserInput();
    }
  }

  private void run() {
    if (login()){
      System.out.println("\nSUCESSFUL LOGIN! Hit the enter key to continue.");
      getUserInput();
    describeGlobalSample();
    describeSample();
    querySample();
    }
  }
}
```

C# Sample Code

This section walks through a sample C# client application. The purpose of this sample application is to show the required steps for logging in and to demonstrate the invocation and subsequent handling of several sforce API calls.

This sample application performs the following main tasks:

1. Prompts the user for their salesforce.com user name and password.
2. Calls login to log in to the sforce single login server and, if the login succeeds:
   - Sets the returned `sessionID` into the session header, which is required for session authentication on subsequent API calls.
   - Resets the sforce Web service endpoint to the returned `serverUrl`, which is the server that will be the target of subsequent API calls.

   All client applications that access the sforce Web services API *must* complete the tasks in this step before attempting any subsequent API calls.
3. Calls describeGlobal to retrieve a list of all available objects for the organization's data.
4. Calls describeSObject to retrieve metadata (field list and object properties) for a specified sforce object.
5. Calls query, passing a simple query string (**`"select FirstName, LastName from Contact"`**), and iterating through the returned QueryResult.

In the following sample code, sforce API calls and other significant code is identified in a **bold** font. In addition, note the error handling code that follows each API call.

The following code begins the sample C# client application.

```
using System;

namespace Walkthrough
{
    class WalkthroughSample
    {
```

```
private sforce.SforceService binding;
static private WalkthroughSample walkthroughSample;
[STAThread]
static void Main(string[] args)
{
    walkthroughSample = new WalkthroughSample();
    walkthroughSample.run();
}

public void run()
{
    //Call the login function
    if ( login() )
    {
        //Do a describe global
        describeGlobal();

        //describe an account object
        describeSObject("account");

        //retrieve some data using query
        querySample();
    }
}
```

The `login()` method retrieves the user's login credentials, instantiates a binding stub (which is the main interface to the web service through which all calls are made), and invokes the login call.

For the login call, the initial endpoint is always http(s)//www.salesforce.com. Once the client application has logged in successfully, however, a client application uses the results of the login call to reset the endpoint of the service to the virtual server instance that is servicing our organization's data. After logging in successfully, the sample client application

- instantiates a persistent SOAP header and sets its `sessionId` value to the session ID returned in the LoginResult (`loginResult.sessionId`). This SOAP header is included on all subsequent calls that are made with the binding.
- changes the server endpoint (`binding.URL`) to the URL specified in the returned LoginResult object (`loginResult.serverUrl`).

```
private bool login()
{
    //Get the user name and password from the console
    Console.Write("Username: ");
    string username = Console.ReadLine();
    Console.Write("Password: ");
    string password = Console.ReadLine();

    //create a new instance of the web service proxy class
    binding = new sforce.SforceService();

    try
    {
        //execute the login placing the results
        //in a LoginResult object
      sforce.LoginResult loginResult = binding.login(username, password);

        //set the session id header for subsequent calls
        binding.SessionHeaderValue = new sforce.SessionHeader();
        binding.SessionHeaderValue.sessionId = loginResult.sessionId;
```

```
            //reset the endpoint url to that returned from login
            binding.Url = loginResult.serverUrl;
            return true;
        }
        catch (Exception ex)
        {
            //Login failed, report message then return false
            Console.WriteLine("Login failed with message: " + ex.Message);
            return false;
        }
    }
```

The `describeGlobal` method determines the objects that are available to the logged in user. It invokes a describeGlobal call, which returns all of the objects that are visible to the logged in user (see Factors that Affect Data Access on page 23). This call should not be made more than once per session, as the data returned from the call likely does not change frequently. The DescribeGlobalResult is simply echoed to the console.

```
    private void describeGlobal()
    {
        //The describe global will return an array of object names that
        //are available to the logged in user
        sforce.DescribeGlobalResult dgr = binding.describeGlobal();

        Console.WriteLine("\nDescribe Global Results:\n");

        //Loop through the array echoing the object names to the console
        for (int i=0;i<dgr.types.Length;i++)
        {
            Console.WriteLine(dgr.types[i]);
        }
        Console.WriteLine("\n\nHit enter to continue...");
        Console.ReadLine();
    }
```

The `describeSObject` method illustrates the type of metadata information that can be obtained for each object available to the user. The sample client application executes a describeSObject call on a given object and then echoes the returned metadata information to the console. Object metadata information includes permissions, field types and lengths, and available values for picklist fields and types for `referenceTo` fields.

```
    private void describeSObject(string objectType)
    {
        //Call the describeSObject passing in the object type name
        sforce.DescribeSObjectResult dsr =
binding.describeSObject(objectType);

        //The first properites we will echo are on the object itself
        //First we will output some Descriptive info on the object
        Console.WriteLine("\n\nObject Name: " + dsr.name);
        if (dsr.custom) Console.WriteLine("Custom Object");
        if (dsr.label != null) Console.WriteLine("Label: " + dsr.label);

        //now the permissions on the object
        if (dsr.activateable) Console.WriteLine("Activateable");
        if (dsr.createable) Console.WriteLine("Createable");
        if (dsr.deletable) Console.WriteLine("Deleteable");
        if (dsr.queryable) Console.WriteLine("Queryable");
        if (dsr.replicateable) Console.WriteLine("Replicateable");
        if (dsr.retrieveable) Console.WriteLine("Retrieveable");
```



```
if (dsr.searchable) Console.WriteLine("Searchable");
if (dsr.undeletable) Console.WriteLine("Undeleteable");
if (dsr.updateable) Console.WriteLine("Updateable");

//Now we will retrieve meta-data about each of the fields
for (int i=0;i<dsr.fields.Length;i++)
{
    //Create field object for readability
    sforce.Field field = dsr.fields[i];

    //Echo some useful information
    Console.WriteLine("Field name: " + field.name);
    Console.WriteLine("\tField Label: " + field.label);
    //This next property indicates that this
    //field is searched when using
    //the name search group in SOSL
    if (field.nameField)
        Console.WriteLine("\tThis is a name field.");
    if (field.restrictedPicklist)
        Console.WriteLine("This is a RESTRICTED picklist field.");
    Console.WriteLine("\tType is: " + field.type.ToString());
    if (field.length > 0)
        Console.WriteLine("\tLength: " + field.length);
    if (field.scale > 0)
        Console.WriteLine("\tScale: " + field.scale);
    if (field.precision > 0)
        Console.WriteLine("\tPrecision: " + field.precision);
    if (field.digits > 0)
        Console.WriteLine("\tDigits: " + field.digits);
    if (field.custom)
        Console.WriteLine("\tThis is a custom field.");

    //Output the permission on this field.
    if (field.nillable) Console.WriteLine("\tCan be nulled.");
    if (field.createable) Console.WriteLine("\tCreateable");
    if (field.filterable) Console.WriteLine("\tFilterable");
    if (field.updateable) Console.WriteLine("\tUpdateable");

    //If this is a picklist field, we will show the values
    if (field.type.Equals(sforce.fieldType.picklist))
    {
       Console.WriteLine("\tPicklist Values");
          for (int j=0;j<field.picklistValues.Length;j++)
          Console.WriteLine("\t\t" + field.picklistValues[j].value);
    }

    //If this is a foreign key field (reference),
    //we will show the values
    if (field.type.Equals(sforce.fieldType.reference))
    {
        Console.WriteLine("\tCan reference these objects:");
        for (int j=0;j<field.referenceTo.Length;j++)
           Console.WriteLine("\t\t" + field.referenceTo[j]);
    }
    Console.WriteLine("");

}

Console.WriteLine("\n\nHit enter to continue...");
Console.ReadLine();
```



```
        }
```

The `querySample()` method executes a query by invoking the query call, passing a simple query string (**`"select FirstName, LastName from Contact"`**), and iterating through the returned QueryResult.

```
        private void querySample()
        {
            //The results will be placed in qr
            sforce.QueryResult qr = null;

            //We are going to limit our return batch size to 3 items
            binding.QueryOptionsValue = new sforce.QueryOptions();
            binding.QueryOptionsValue.batchSize = 3;
            binding.QueryOptionsValue.batchSizeSpecified = true;

            try
            {
               qr = binding.query("select FirstName, LastName from Contact");
               bool done = false;
               if (qr.size > 0)
               {
                  Console.WriteLine("Logged in user can see " + qr.records.Length
+ " contact records.");
                   while (!done)
                   {
                    Console.WriteLine("");
                    for (int i=0;i<qr.records.Length;i++)
                    {
                       sforce.Contact con = (sforce.Contact)qr.records[i];
                       string fName = con.FirstName;
                       string lName = con.LastName;
                       if (fName == null)
                       Console.WriteLine("Contact " + (i + 1) + ": " + lName);
                       else
                      Console.WriteLine("Contact " + (i + 1) + ": " + fName + " "
+ lName);
                      }
                   if (qr.done)
                   {
                      done = true;
                   }
                   else
                   {
                      qr = binding.queryMore(qr.queryLocator);
                   }
                   }
               }
               else
               {
                  Console.WriteLine("No records found.");
               }
            }
            catch (Exception ex)
            {
                Console.WriteLine("\nFailed to execute query succesfully, error
message was: \n" + ex.Message);
            }
            Console.WriteLine("\n\nHit enter to exit...");
            Console.ReadLine();
```



```
            }
        }
}
```

CHAPTER 2: Basic Concepts

This topic describes the basic concepts you need to understand before building client applications that use the sforce Web services API. It includes the following topics:

- sforce API Calls
- sforce API Objects
- Security in the sforce API
- Error Handling in the sforce API

For additional conceptual information, see Other Concepts on page 249.

SFORCE API CALLS

The sforce API calls, summarized in the following table, represent specific operations that your client applications can invoke at run-time to perform certain tasks.

Table 1: sforce API Calls

Task / Call	Description
Login / Client Session	
login	Logs in to the sforce server and starts a client session.
Querying Data	For details, see sforce Object Query Language (SOQL) on page 28.
query	Executes a query against the specified object and returns data that matches the specified criteria.
queryMore	Retrieves the next batch of objects from a query.
Search & Retrieval	
retrieve	Retrieves one or more objects based on the specified object ID(s).
search	Searches for objects based on keywords. For details, see sforce Object Search Language (SOSL) on page 33.
Modifying Data	
create	Adds one or more new individual objects to your organization's data.
update	Updates one or more existing objects in your organization's data.
delete	Deletes one or more individual objects from your organization's data.
Object Metadata	
describeGlobal	Retrieves a list of available objects for your organization's data.
describeSObject	Retrieves the next batch of objects from a query.
Data Replication	For details, see Data Replication on page 40.

Table 1: sforce API Calls (Continued)

Task / Call	Description
getDeleted	Retrieves the IDs of individual objects that have been deleted since the specified time for the specified object.
getUpdated	Retrieves the IDs of individual objects that have been updated since the specified time for the specified object.
Utilities	For details, see sforce Utility API Calls on page 94.
getServerTimestamp	Retrieves the current system timestamp from the sforce Web service.
getUserInfo	Retrieves personal information for the user associated with the current session.
resetPassword	Changes a user's password to a server-generated value.
setPassword	Sets the specified user's password to the specified value.

All sforce API calls are synchronous requests made to the sforce Web service. Any changes to your salesforce.com data are committed automatically. For more information, see sforce API Calls on page 22, including the following topics:

- Concepts on page 22
- sforce Object Query Language (SOQL) on page 28
- sforce Object Search Language (SOSL) on page 33
- Data Replication on page 40
- List of sforce API Calls on page 43

SFORCE API OBJECTS

In the sforce API, *objects* are data entities that represent your organization's information. The sforce Web services API interacts with your organization's data via *objects*, which are programmatic representations of your organization's salesforce.com data. *Object properties* represent fields in those data entities, and client applications set or retrieve data values via these properties.

For more information, see sforce Objects on page 101, including the following topics:

- Concepts on page 101
- List of sforce Objects on page 110
- Allowed API Calls on sforce Objects on page 113
- Individual sforce API objects, beginning with Account on page 115

SECURITY IN THE SFORCE API

Client applications that access your organization's sensitive salesforce.com data are subject to the same security protections that are used in the salesforce.com user interface.

User Authentication

Client applications must log in using valid credentials for a salesforce.com account. The sforce server authenticates these credentials and, if valid, provides the client application with:

- a sessionID that must be set into the session header so that all subsequent API calls are authenticated

- an URL for the sforce Web service that will service the client application's Web service requests

Profile Configuration

Permissions to access data and invoke API calls are governed by the profile associated with the salesforce.com account under which the client application logs in. The organization's salesforce.com administer controls access by configuring the profile settings and assigning users to that profile (**Setup** | **Manage Users** | **Profiles**). Client applications can query or update only those objects and fields to which they have appropriate access.

Note The sforce Web services WSDL files return all available objects and fields for an organization.

Sharing

In the salesforce.com user interface, the concept of *sharing* refers to the act of granting access to a user or group to view and/or edit a record owned by another user, if the default organization access levels do not otherwise permit such access. All sforce API calls respect the Sharing model.

Note In the sforce API, you can create and modify some sharing objects (AccountShare and OpportunityShare). However, to configure sharing rules for these sharing objects, you must do so via the salesforce.com user interface.

Implicit Restrictions for Objects and Fields

Certain sforce objects can be created or deleted only in the salesforce.com user interface. Other sforce objects are read-only—client applications cannot create, update, or delete such objects. Similarly, certain fields within some sforce objects can be specified on create but not on update. Other fields are read-only—client applications cannot specify field values in create or update calls. For more information, see the object description in sforce Objects on page 101.

Error Handling in the sforce API

The sforce API calls return error data that your client application can use to identify and resolve run-time errors. If an error occurs during the invocation of an API call, then the sforce Web service throws an exception and returns an `ApiFault` with an associated ExceptionCode and error message text that provide additional information about the error.

For more information about errors, see the following topics:

- Error on page 25
- ExceptionCode on page 25
- ApiFault on page 27
- List of APIFault Codes on page 28

CHAPTER 3: sforce API Calls

This topic describes the sforce Web services API calls that your client applications can invoke to retrieve and change your organization's information. It contains the following sections:

- Concepts
- sforce Object Query Language (SOQL)
- sforce Object Search Language (SOSL)
- Data Replication
- List of sforce API Calls

The rest of this topic describes most of the sforce API calls in detail. Some of the sforce API calls are described in sforce Utility API Calls on page 94.

CONCEPTS

This section describes the following concepts:

- About sforce API Calls
- Characteristics of sforce API Calls
- Factors that Affect Data Access
- Typical API Call Sequence
- Start By Logging In to the sforce Server
- Core Data Objects
- List of APIFault Codes

About sforce API Calls

The sforce API calls represent specific operations that your client applications can invoke at run-time to perform certain tasks. For example, you can query your organization's data; add, update, and delete information; obtain metadata about your organization's data; and run utilities to perform administration tasks. For detailed information about sforce utility API calls, see sforce Utility API Calls on page 94.

Characteristics of sforce API Calls

All sforce API calls are:

- **Service requests and responses**—Your client application prepares and submits a service request to the sforce Web service, the sforce Web service processes the request and returns a response, and the client application handles the response as appropriate.
- **Synchronous**—Once the API call is invoked, your client application waits until it receives a response from the service. Asynchronous calls are not supported.
- **Committed automatically**—Every operation that writes to a salesforce.com table is committed automatically. This is analogous to the AUTOCOMMMIT setting in SQL. For create, update, and delete calls that attempt to write to multiple rows in a table, the write operation for *each* row is treated as a *separate transaction*. For example, if a client application attempts to create two new accounts, they're created using mutually exclusive insert operations that succeed or fail individually, not as a group.

Factors that Affect Data Access

When using the sforce API, the following factors affect access to your organization's data:

- Depending on which WSDL you are using:
  - **Enterprise WSDL**: The generated enterprise.wsdl file contains all of the objects that are available to your organization. A client application can access, via the sforce API, objects that are defined in the enterprise.wsdl file that you are using.
  - **Partner WSDL**: When using the generated Partner WSDL file, a client application can access objects that are returned in the describeGlobal call.

  For more information, see Generating the WSDL File for Your Organization on page 4.
- Whether your configured permissions allow access to the data. Your client application logs in as a user to the sforce Web service. The permissions profile associated with that logged in user determine the level of access to specific objects and fields in your organization's information.

  For certain objects, the user profile is configured with one of the following permissions:
  - **Read**—Users can only view objects of this type.
  - **Create**—Users can read and create objects of this type.
  - **Edit**—Users can read and update objects of this type.
  - **Delete**—Users can read, edit, and delete objects of this type.

  These permissions apply to the following objects: Account, Asset, Campaign, Case, Contact, Contract. Document. Lead, Opportunity, Pricebook2, and Product2. Other sforce objects (such as AccountTeamMember), follow sharing on the associated permission-assigned object (such as the Account record). Similarly, a Partner depends on the permissions in the associated Account.

  User permissions do not affect field-level security or sharing. If field-level security specifies that a field is hidden, users with the "Read" permission can view only those fields that are not hidden on the record. In addition, users with "Read" permissions can view only those records that sharing settings allow. The one exception is the "Edit Read Only Fields" permission, which gives users the ability to edit fields marked as read only via field-level security.

  The sforce API respects object-level and field-level security configured in the salesforce.com user interface. You can access objects and fields only if the security settings in the logged-in user's personal profile permit such access. For example, fields that are not visible to a given user are not returned in a query or describeSObject call. For more information, see Security in the sforce API on page 20.
- Whether the sharing model of the logged-in user allows access to the data. For most API calls, data that is outside of the logged-in user's sharing model is *not* returned.
- Whether a given object is configured to be accessible via certain API calls. For example, to create an object via the create call, its object must be configured as createable (`createable=True`). To determine what operations are allowed on a given object, your client application can invoke the describeSObject call on the object and inspect the following properties in the DescribeSObjectResult: `createable` (create call), `updateable` (update call), `deletable` (delete call), `queryable` (query call), `retrieveable` (retrieve call), `searchable` (search call), and `replicateable` (getUpdated and getDeleted calls).
- Whether a particular change would compromise the referential integrity of your organization's salesforce.com data. For example:
  - ID values in reference fields (see Reference Field Type on page 104) are validated in create and update calls.
  - If a client application deletes an object instance, then its children are automatically deleted as well (cascading deletes). For example, if a client application deletes an opportunity, then any associated opportunity line items are also deleted.
  - A client application cannot delete an object instance if it is referenced by another object instance. For example, a client application cannot delete a particular Account instance if the `AccountId` field in an Opportunity instance references the ID of that account.

- Whether a given field in an sforce object can be updated or not. For example, read-only fields cannot be changed in create or update calls.
- Whether a given feature is used by your organization. For example, the `RecordTypeId` field will appear in your WSDL file only if at least one record type is configured for your organization in the salesforce.com user interface.
- Rules for custom objects—such as fields that are configured in the salesforce.com user interface to be required (not null) or unique—are not implicitly enforced via the sforce API.
- Ownership changes to one object instance do not automatically cascade to other object instances. For example, if ownership changes for a given Account, ownership does not then automatically change for any Contract associated with that Account—each ownership change must be made separately and explicitly by the client application.
- Certain features that are configurable in the salesforce.com user interface are not accessible or implicitly enforced via the sforce API. For example:
  - Layouts can specify whether a given field is required, but the sforce API does not enforce such layout-specific field restrictions or validations in create and update calls. It is up to the client application to enforce any such constraints, if applicable.
  - Record types can control which picklist values can be chosen in a given record and which which page layouts users with different profiles can see. However, such rules that are configured and enforced in the salesforce.com user interface are not enforced in the sforce API. For example, the sforce Web service will not validate whether the value in a picklist field is allowed per any record type restrictions associated with the profile of the logged in user. Similarly, the sforce Web service will not prevent a client application from adding data to a particular field simply because that field does not appear in a layout associated with the profile of the logged in user.

  If any such constraints are required, it is up to business logic in the client application to enforce them explicitly.

Typical API Call Sequence

For each sforce API call, your client application typically:

- Prepares the request by defining request parameters, if applicable
- Invokes the call, which passes the request with its parameters to the sforce Web service for processing
- Receives the response (synchronously) from the sforce Web service
- Handles the response, either by processing the returned data (for a successful invocation) or by handling the error (for a failed invocation).

Start By Logging In to the sforce Server

Before invoking any other sforce API calls, a client application must first invoke the login call to establish a session with the sforce server, set the returned server URL as the target server for subsequent API requests, and set the returned session ID in the SOAP header to provide server authorization for subsequent API requests. For more information, see login on page 76 and Sample Code Walkthrough on page 6.

Core Data Objects

Many calls in the sforce API use the following data objects:

- ID
- sObject
- Error
- ExceptionCode
- ApiFault

ID

Almost all objects in the sforce API have an associated ID, which is a string (18 alphanumeric characters in length) that uniquely identifies an individual object. An ID is analogous to a primary or foreign key field in a database table. When you create a new object, the sforce Web service generates an `ID` value for the object, ensuring that it is properly formatted and unique within your organization's data. Thereafter, you can refer to the object by its unique ID in subsequent sforce API calls. For more information, see ID Fields on page 106.

Note In Visual Studio, ID objects are treated as strings. In Java, ID objects are treated as objects.

sObject

An `sObject` represents an sforce object, such as an individual Account or Campaign. For a complete list of sforce objects, see Chapter 5: sforce Objects on page 101.

An `sObject` has the following properties:

Name	Type	Description
fieldsToNull	string[]	Array of one or more field names whose value you want to explicitly set to `null`. Used only with the update call. Ensures that, for this `sObject`, the value in the specified fields will be set to `null`. You can specify only those fields that you can update and that are nillable. For example, specifying an ID field or required field results in a run-time error.
ID	ID	Unique ID for this individual object. For the create call, this value is null. For all other sforce API calls, this value must be specified.

Error

An `Error` contains information about an error that occurred during an sforce API call (create, update, or delete only). For more information, see Error Handling in the sforce API on page 21.

An `Error` has the following properties:

Name	Type	Description
StatusCode	ExceptionCode (enum)	Status code that characterizes the error.
message	string	Error message text.
fields	string[]	Reserved for future use.

ExceptionCode

An `ExceptionCode` contains information about an ApiFault that occurred during an sforce API call. For more information, see Error Handling in the sforce API on page 21.

The following list of `ExceptionCode` values is defined in your WSDL file:

ExceptionCode	Description
API_CURRENTLY_DISABLED	API functionality is temporarily down due to a server problem.
API_DISABLED_FOR_ORG	Organization is not enabled for use of the API. A representative from the organization must contact salesforce.com to enable API access.
EXCEEDED_ID_LIMIT	Too many IDs were requested in a retrieve call.
EXCEEDED_LEAD_CONVERT_LIMIT	Too many IDs were sent to a convertLead call.
EXCEEDED_MAX_SIZE_REQUEST	Size of the message sent to the sforce Web service exceeded 50MB.
EXCEEDED_QUOTA	Organization storage limits have been exceeded during a create call.
EXCEEDED_RATE_LIMIT	Client sent concurrent API requests and the original request has been terminated.
FUNCTIONALITY_NOT_ENABLED	Functionality has been temporarily disabled. Other calls may continue to work.
INSUFFICIENT_ACCESS	Logged-in user does not have sufficient access to perform this operation.
INVALID_ASSIGNMENT_RULE	Specified AssignmentRuleHeader value is not valid.
INVALID_BATCH_SIZE	Batch size specified in the query options is out of the supported range.
INVALID_CLIENT	Client is invalid.
INVALID_FIELD	Specified field name is invalid.
INVALID_LOGIN	Invalid login credentials.
INVALID_NEW_PASSWORD	New password does not conform with the organization's password settings.
INVALID_OPERATION_WITH_EXPIRED_PASSWORD	Client application that logged on with an expired password invoked an invalid call (only setPassword is allowed).
INVALID_QUERY_FILTER_OPERATOR	An operator used in the query filter clause is invalid, at least for that field.
INVALID_QUERY_LOCATOR	Specified `queryLocator` parameter in a queryMore call is invalid.
INVALID_REPLICATION_DATE	Date for replication is out of range, such as before the salesforce.com 30 day limit (previous to the current date) for keeping deleted date, or before the organization was created.
INVALID_SEARCH	Invalid syntax or grammar specified in the search call. See sforce Object Search Language (SOSL) on page 33.
INVALID_SEARCH_SCOPE	Specified search scope is invalid.

ExceptionCode	Description
INVALID_SESSION_ID	Specified sessionID is invalid or has expired. You should log in again to generate a new session.
INVALID_TYPE	Specified sObject type is invalid.
LOGIN_DURING_RESTRICTED_DOMAIN	User is restricted from logging in from this IP address.
LOGIN_DURING_RESTRICTED_TIME	User is restricted from logging in during this time period.
MALFORMED_QUERY	Specified query string is not valid. For example, the query string was longer than 10,000 characters.
MALFORMED_SEARCH	Specified search string is invalid. For example, the search string was longer than 10,000 characters.
ORG_LOCKED	Organization has been locked. You must contact salesforce.com to re-enable the organization.
PASSWORD_LOCKOUT	User has attempted multiple invalid logins and has been locked out. The user must contact the organization administrator to re-enable the account.
QUERY_TIMEOUT	Query timed out. For more information, see sforce Object Query Language (SOQL) on page 28.
SERVER_UNAVAILABLE	A server that is necessary for this call, such as the search call, is currently down. Other types of requests might still work.
TRIAL_EXPIRED	Organization is a trial organization that has reached its expiration date. A representative from the organization must contact salesforce.com to re-enable the organization.
UNKNOWN_EXCEPTION	Server encountered an internal error. You should report this problem to salesforce.com.
UNSUPPORTED_API_VERSION	WSDL file represents an unsupported version of the sforce API.
UNSUPPORTED_CLIENT	This version of the client is no longer supported.

ApiFault

If an error occurs during the invocation of an API call, the sforce Web service throws an exception and returns an `ApiFault` with an associated ExceptionCode and error message text that provide additional information about the error. For more information about `ApiFault`, see List of APIFault Codes on page 28. Fault codes are of type `ApiFault`, which has the following properties.

Name	Type	Description
exceptionCode	ExceptionCode	Exception code.
exceptionMessage	string	Error message text.

List of APIFault Codes

The following table lists the ApiFault codes that the sforce Web service returns if an error occurs when processing a service request.

Table 2: Fault Codes for sforce API Calls

Fault	Description
`LoginFault`	Error occurred during the login call.
`InvalidSObjectFault`	Invalid sObject in a describeSObject, create, update, retrieve, or query call.
`InvalidFieldFault`	Invalid field in a retrieve or query call.
`MalformedQueryFault`	Problem in the `queryString` passed in a query call.
`InvalidQueryLocatorFault`	Problem in the `queryLocator` passed in a queryMore call.
`MalformedSearchFault`	Problem in the `searchString` passed in a search call.
`InvalidIdFault`	Specified ID was invalid in a setPassword or resetPassword call.
`UnexpectedErrorFault`	Unexpected error occurred. The error is not associated with any other ApiFault.

sforce Object Query Language (SOQL)

You use the sforce Object Query Language (SOQL) to construct simple but powerful query strings for the `queryString` parameter in the query call. Similar to the SELECT command in SQL, SOQL allows you to specify the source object (such as Account), a list of fields to retrieve, and conditions for selecting rows in the source object. This topic includes:

- SOQL Syntax
- conditionExpression Syntax
- fieldExpression Syntax
- Logical Operators
- Changing the Batch Size in Queries
- Querying Multi-Select Picklists

Note

SOQL does not support all advanced features of the SQL SELECT command. For example, you cannot use SOQL to perform join operations, use wildcards in field lists, use calculation expressions, or specify an ORDERBY clause to sort rows in the result set.

SOQL Syntax

SOQL uses the following syntax:

`select` *`fieldList`* `from` *`objectType`* `[where` *`conditionExpression`*`]`

where:

Syntax	Description
fieldList	Specifies a list of one or more fields, separated by commas, that you want to retrieve from the specified *object*. You must specify valid field names and must have read-level permissions to each specified field. The *fieldList* defines the ordering of fields in the query results.
objectType	Specifies the type of sforce object that you want to query. You must specify a valid sforce object and must have read-level permissions to that object. For a list of valid objects, see List of sforce Objects on page 110.
conditionExpression	Determines which rows in the specified *object* to retrieve. If unspecified, the query retrieves all rows in the *object*. See conditionExpression Syntax on page 29 for the appropriate syntax.

NOTE

SOQL statements cannot exceed 10,000 characters. For SOQL statements that exceed this maximum length, the sforce Web service returns an ExceptionCode of `MALFORMED_QUERY`; no result rows are returned.

conditionExpression Syntax

The *conditionExpression* uses the following syntax:

fieldExpression [*logicalOperator fieldExpression2*][*logicalOperator fieldExpression3*]...

- You can use parentheses to define the order in which fieldExpressions are evaluated. For example, the following expression is True if *fieldExpression1* is True and *either* *fieldExpression2* or *fieldExpression3* are True.

  fieldExpression1 AND (*fieldExpression2* OR *fieldExpression3*)

- However, the following expression is True if either *fieldExpression3* is True or both *fieldExpression1* and *fieldExpression2* are True.

  (*fieldExpression1* AND *fieldExpression2*) OR *fieldExpression3*

- Client applications must specify parentheses when nesting operators. However, multiple operators of the same type do not need to be nested.

See fieldExpression Syntax on page 29 for the syntax of *fieldExpression*s. See Logical Operators on page 31 for the valid logical operators.

fieldExpression Syntax

A *fieldExpression* uses the following syntax:

fieldName comparisonOperator value

where:

Syntax	Description
fieldName	The name of a field in the specified *object*. Use of single or double quotes around the name will result in an error. You must have at least read-level permissions to the field. It can be any field—it does not need to be a field in the *fieldList*.
comparisonOperator	One of the comparison operators listed in Comparison Operators on page 30.
value	A value, enclosed in *single quotes* (double quotes result in an error), used to compare with the value in *fieldName*. You must supply a value whose data type matches the field type of the specified field. You must supply a native value—other field names or calculations are not permitted. For date values, use the formatting listed in Date Formats on page 31.

Comparison Operators

A *fieldExpression* uses the following *comparisonOperator*s:

Operator	Name	Description
=	Equals	Expression is True if the value in the specified *fieldName* equals the specified *value* in the expression.
!=	Not equals	Expression is True if the value in the specified *fieldName* does *not* equal the specified *value*.
<	Less than	Expression is True if the value in the specified *fieldName* is less than the specified *value*.
<=	Less or equal	Expression is True if the value in the specified *fieldName* is less than, or equals, the specified *value*.
>	Greater than	Expression is True if the value in the specified *fieldName* is greater than the specified *value*.
>=	Greater or equal	Expression is True if the value in the specified *fieldName* is greater than, or equal to, the specified *value*.

Operator	Name	Description

like	Like	Expression is True if the value in the specified *fieldName* matches the characters of the text string in the specified *value*.
		The *like* operator in SOQL is similar to the same operator in SQL; it provides a mechanism for matching partial text strings and includes support for wildcards.
		• The % and _ wildcards are supported for the *like* operator.
		  • The % wildcard matches zero or more characters.
		  • The _ wildcard matches exactly one character.
		• The text string in the specified *value* must be enclosed in single quotes.
		• The *like* operator is supported for string fields only (see String Field Type on page 103).
		• The *like* operator performs a case-insensitive match, unlike the case-sensitive matching in SQL.
		• The *like* operator in SOQL does not currently support escaping of special characters such as % or _. The \ (backslash) character should not be used.
		`select AccountId, FirstName, lastname from Contact where lastname like '%appl_%'`
		matches *Appleton*, *Apple*, *Bapple*, but not *Appl*.

includes **excludes**		Applies only to multi-select picklists. See Querying Multi-Select Picklists on page 32.

Date Formats

A *fieldExpression* uses the following date formats (milliseconds and time zone are optional):

Use	Format Syntax	Example
Date only	`YYYY-MM-DD`	`1999-01-01`
Date and time	`YYYY-MM-DDThh:mm:ss`	`1999-01-01T24:01:01`
Date, time, and milliseconds	`YYYY-MM-DDThh:mm:ss.MILLIS`	`1999-01-01T24:01:01.001`
Date, time, milliseconds, and time zone offset	• `YYYY-MM-DDThh:mm:ss.millis+hh:mm` • `YYYY-MM-DDThh:mm:ss.millis-hh:mm` • `YYYY-MM-DDThh:mm:ss.millisZ`	• `1999-01-01T23:01:01.001+01:00` • `1999-01-01T23:01:01.001-08:00` • `1999-01-01T23:01:01.001Z`

The zone offset is always from UTC. For more information, see:

- http://www.w3.org/TR/xmlschema-2/#isoformats
- http://www.w3.org/TR/NOTE-datetime

Note

For a *fieldExpression* that uses date formats, the date is not enclosed in single quotes. No quotes should be used around the date. For example:

```
select Id from Account where CreatedDate > 2003-10-29T11:30:00Z
```

Logical Operators

A *logicalOperator* is used on one or more *fieldExpression*s. A *logicalOperator* is one of the following values:

Operator	Syntax	Description
and	*fieldExpressionX* and *fieldExpressionY*	True only if both *fieldExpressionX* and *fieldExpressionY* are True.
or	*fieldExpressionX* or *fieldExpressionY*	True if either *fieldExpressionX* or *fieldExpressionY* is True.
not	not *fieldExpressionX*	True if *fieldExpressionX* is False.

Changing the Batch Size in Queries

By default, the batch size for the number of records returned in a query or queryMore call is set to 500. Client applications can change this setting by specifying the batch size in the QueryOptions portion of the SOAP header before invoking the query call. The maximum batch size is 2,000.

Note

The batch size will be no more than 200 if the SOQL statement selects two or more custom fields of type long text. This is to prevent large SOAP messages from being returned.

The following sample Java (Axis) code demonstrates setting the batch size to three (3) records.

```
_QueryOptions qo = new _QueryOptions();
qo.setBatchSize(new Integer(3));
binding.setHeader(new SforceServiceLocator().getServiceName().getNamespaceURI(),
"QueryOptions", qo);
```

The following sample C# (.NET) code demonstrates setting the batch size to three (3) records.

```
binding.QueryOptionsValue = new QueryOptions();
binding.QueryOptionsValue.batchSize = 3;
binding.QueryOptionsValue.batchSizeSpecified = true;
```

Querying Multi-Select Picklists

Client applications use a specific syntax for querying multi-select picklists (in which multiple items can be selected).

Supported Operators

The following operators are supported for querying multi-select picklists.

Operator	Description
=	Equals the specified string.
!=	Does not equal the specified string.
includes	Includes (contains) the specified string.
excludes	Excludes (does not contain) the specified string.

Semicolon Character

A semicolon is used as a special char to specify a union (AND). For example the following notation:

```
'AAA;BBB'
```


means 'AAA' and 'BBB'.

Examples

In the following example SOQL notation:

```
MSP1__c = 'AAA;BBB'
```

the query filters on values in the **MSP1__c** field that are equal to AAA and BBB selected (exact match).

In the following example SOQL notation:

```
MSP1__c includes ('AAA;BBB', 'CCC')
```

the query filters on values in the **MSP1__c** field that contains both AAA *and* BBB selected, *or* contains CCC selected. A match will result on any field value that contains 'AAA' and 'BBB' OR any field that contains 'CCC'. For example, the following will be matched:

- matches with 'AAA;BBB':

  'AAA;BBB'

  'AAA;BBB;DDD'

- matches with 'CCC':

  'CCC'

  'CCC;EEE'

  'AAA;CCC'

  'BBB;CCC'

sforce Object Search Language (SOSL)

The sforce Object Search Language (SOSL) is used to construct simple but powerful text searches for the search call. This topic describes SOSL syntax and usage.

About SOSL

SOSL allows you to specify the text expression, the scope of fields to search, the list of objects and fields to retrieve, and the maximum number of objects to return. You pass the entire SOSL expression in the `searchString` parameter of the search call.

Like the sforce Object Query Language (SOQL), SOSL allows you programmatically search your organization's salesforce.com data for specific information. You can search all objects—including custom objects—to which you have access. The sforce Web service executes the search within the specified scope and returns to you only the information that is available to you based on the user permissions under which your application has logged in.

Designing Efficient Text Searches

When designing text searches, make sure that your text searches are designed with efficiency and performance in mind. A text search should be sufficiently comprehensive to gather all the information you need but also properly focused to search only where you want to look. SOSL syntax allows you to define the search scope, both in the types of columns to search (IN clause) and the objects to search (RETURNING clause). For examples, see the discussions of IN SearchGroup on page 37 and RETURNING FieldSpec on page 38.

Testing Searches in the salesforce.com Interface

Before coding your text search programmatically, you can *test* your text search syntax using the sidebar Search or Advanced Search features in salesforce.com interface. Programmatic searches are most similar (but not identical) to the Advanced Search feature—programmatic searches are

more exacting, requiring you to structure your searches carefully, while the Advanced Search feature is prestructured and is more forgiving to the end user.

Search Scope

The search call searches most objects (including custom objects) and text fields to which you have access. It does *not* search the following objects and fields:

- Any objects (such as picklists) that are defined as not searchable (`searchable=false`). To determine whether a given object is searchable, your application can invoke the describeSObject call on the object and inspect the `searchable` property in the DescribeSObjectResult.
- Number, date, checkbox, or textarea fields. To search for such information, use the query call instead.
- Attachment objects associated with certain objects, such as Account, Contact, or Opportunity objects.

Note

The search call does not provide specialized search features such as synonym matching or stop words.

SOSL Syntax

SOSL uses the following syntax:

```
FIND {SearchQuery}
[ IN SearchGroup ]
[ RETURNING FieldSpec ]
[ LIMIT n ]
```

where:

Syntax	Description
`FIND {SearchQuery}`	Required. Specifies the text (words or phrases) to search for. The *`SearchQuery`* must be delimited with curly braces.
`IN SearchGroup`	Optional. Scope of fields to search. One of the following values: • `ALL FIELDS` • `NAME FIELDS` • `EMAIL FIELDS` • `PHONE FIELDS` If unspecified, then the default is `ALL FIELDS`. **Note:** You specify the list of objects to search in the `RETURNING` *`FieldSpec`* clause.
`RETURNING FieldSpec`	Optional. Information to return in the search result. List of one or more objects and, within each object, list of one or more fields. If unspecified, then the search results contain the IDs of all objects found.
`LIMIT n`	Optional. Maximum number of rows to return. If unspecified, then the default is the logical maximum limit of 200 rows.

NOTE

SOSL statements cannot exceed 10,000 characters. For SOSL statements that exceed this maximum length, the sforce Web service returns an ExceptionCode of `MALFORMED_SEARCH`; no result rows are returned.

FIND {SearchQuery}

The required `FIND` clause allows you to specify the word or phrase to search for. A search query includes the literal text (single word or a phrase surrounded by double quotes) to search for and, optionally, Wildcards, and logical and grouping (parentheses) Operators. Searches are evaluated from left to right and are conducted in Unicode (UTF-8) encoding. Text searches are case-*insensitive*. For example, searching for `Customer`, `customer`, or `CUSTOMER` all return the same results.

Note that special types of text expressions (such as macros, functions, or regular expressions) that are evaluated at run time are *not* allowed in the `FIND` clause.

Note

The `SearchQuery` *must* be delimited with curly braces. This is needed to unambiguously distinguish the search expression from other clauses in the text search.

Single Words and Phrases

A `SearchQuery` contains two types of text:

- **Single Word**—A single word, such as `test` or `hello`. Words in the `SearchQuery` are delimited by spaces, punctuation, and changes from letters to digits (and vice-versa). Words are always case insensitive. In Chinese, Japanese, and Korean (CJK), words are also delimited by pairs of CJK-type characters.
- **Phrase**—A collection of words and spaces surrounded by double quotes such as `"john smith"`. Multiple words can be combined together with logical and grouping Operators to form a more complex query. Certain keywords (`AND`, `OR`, and `AND NOT`) must be surrounded in double quotes if you want to search for those words.

Wildcards

You can specify the following wildcard characters to match text patterns in your search:

Wildcard	Description
*	Use an asterisk (*) to match one or more characters at the middle or end of your search term. Do *not* use the asterisk at the beginning of a search term. If you are searching for a literal asterisk in a word or phrase, then escape the asterisk (precede it with the `\` character). For example, a search for `john*` finds items that start with variations on the term `john`, such as, `johnson` or `johnny`. A search for `ma*` finds items with `mary` or `marty`.
?	Use a question mark (?) to match one character at the middle or end of your search term. Do *not* use the question mark wildcard at the beginning of a search term. For example, a search for `jo?n` finds items with the word `john` or `joan`.

When using wildcards, considering the following issues:

- The more focused your wildcard search, the faster the search results are returned, and the more likely the results will reflect your intention. For example, to search for all occurrences of the word `prospect` (or `prospects`, the plural form), it is more efficient to specify `prospect*` in the search string than to specify a less restrictive wildcard search (such as `prosp*`) that could return extraneous matches (such as `prosperity`).
- Tailor your searches to find all variations of a word. For example, to find `property` and `properties`, you would specify `propert*`.
- Punctuation is indexed. To find `*` or `?` inside a phrase, you *must* enclose your search string in quotation marks and you *must* escape the special character. For example, `"where are`

you\?" finds the phrase where are you?. The escape character (\) is required in order for this search to work correctly.

Operators

You can use the following special operators to focus your text search.

Operator	Description
" "	Use quotation marks around search terms to find an exact phrase match. This can be especially useful when searching for text with punctuation. For example, "acme.com" finds items that contain the exact text acme.com. A search for "monday meeting" finds items that contain the exact phrase monday meeting.
Logical Operators	Note that logical operators are case-insensitive. For example, specifying AND, and, or And returns the same results.
AND	Finds items that match *all* of the search terms. For example, john AND smith finds items with *both* the word john and the word smith. If an operator is not specified, then this is the default operator.
OR	Finds items with at least one of the search terms. For example, john OR smith finds items with *either* john or smith, or *both* words.
AND NOT	Finds items that do not contain the search term. For example, john AND NOT smith finds items that have the word john but not the word smith.
Grouping Operators	
( )	Use parentheses around search terms in conjunction with logical operators to focus your search. For example, you can search for: • ("Bob" and "Jones") OR ("Sally" and "Smith")—searches for *either* Bob Jones *or* Sally Smith. • ("Bob") and ("Jones" OR "Thomas") and Sally Smith—searches for Bob Jones or Bob Thomas *and* Sally Smith.

Reserved Characters

The following characters are reserved for current and future use:

```
& | ! ( ) { } [ ] ^ " ~ * ? : \ ' + -
```

In the current version, the following special characters are used:

```
* ? ( ) "
```

Reserved characters, if specified in a text search, must be *escaped* (preceded by the backslash \ character) in order to be properly interpreted. This is true even if the SearchQuery is enclosed in double quotes. An error occurs if you do not precede reserved characters with a backslash. For example, to search for the following text:

```
(1+1):2
```

you must escape the reserved characters in the following manner:

```
\(1\+1\)\:2
```


Example FIND Clauses

Type of Search	Example(s)
Single term	`Find {MyProspect}`
	`Find {mylogin@salesforce.com}`
	`Find {find}`
	`Find {in}`
	`Find {returning}`
	`Find {limit}`
Single phrase	`Find {John Smith}`
Term OR Term	`Find {MyProspect OR MyCompany}`
Term AND Term	`Find {MyProspect AND MyCompany}`
Term AND Phrase	`Find {MyProspect AND "John Smith"}`
Term OR Phrase	`Find {MyProspect OR "John Smith"}`
Complex query using Term/ Phrase using AND/OR	`Find {MyProspect AND "John Smith" OR MyCompany}`
	`Find {MyProspect AND ("John Smith" OR MyCompany)}`
Complex query using Term/ Phrase using AND NOT	`Find {MyProspect AND NOT MyCompany}`
Wildcard Search	`Find {My*}`
Escape sequences	`Find {Why not\?}`
Invalid/Incomplete phrase	`Find {"John Smith}`

IN SearchGroup

The optional `IN` clause allows you to define the types of fields to search. You can specify one of the following values (note that numeric fields are not searchable). If unspecified, the default behavior is to search *all* text fields in searchable objects.

Valid SearchGroup Settings

Scope	Description
`ALL FIELDS`	Search all searchable fields. If the `IN` clause is unspecified, then this is the default setting.
`NAME FIELDS`	Search only name fields. In custom objects, fields that are defined as "Name Field" are searched. In standard and custom objects, name fields have the `nameField` property set to `true` (see the Field array of the `fields` parameter of the DescribeSObjectResult for more information).

Scope	Description
PHONE FIELDS	Search only phone number fields.
EMAIL FIELDS	Search only email fields.

While the IN clause is optional, it is recommended that you specify the search scope unless you need to search all fields. For example, if you're searching only for an email address, you should specify IN EMAIL FIELDS in order to design the most efficient search.

Example IN Clauses

Search Type	Example(s)
No search group	Find {MyProspect}
ALL FIELDS	Find {MyProspect} in ALL FIELDS
EMAIL FIELDS	Find {mylogin@mycompany.com} in EMAIL FIELDS
PHONE FIELDS	Find {MyProspect} in PHONE FIELDS
NAME FIELDS	Find {MyProspect} in NAME FIELDS
Invalid searches	Find {MyProspect} in SIDEBAR FIELDS
	Find {MyProspect} in Accounts

RETURNING FieldSpec

The optional RETURNING clause allows you to specify the information that is returned in the text search result. If unspecified, then the default behavior is to return the IDs of all available objects found (up to the maximum LIMIT n specified in the LIMIT clause). Use the RETURNING clause to restrict the results data that is returned from the search call.

Syntax

RETURNING *ObjectTypeName*[(*FieldList*)][, *ObjectTypeName*[(*FieldList*)], ...

where:

Name	Description
ObjectTypeName	Object to return. If specified, then the search call returns the IDs of all found objects matching the specified object. Must be a valid sObject type. You can specify multiple objects, separated by commas. Objects not specified in the RETURNING clause are *not* returned by the search call.
FieldList	Optional list of one or more fields to return for a given object, separated by commas. If you specify one or more fields, then—in addition to the IDs—the fields are also returned for all found objects. You do not need to specify ID fields, as they are always returned.

Note

The RETURNING clause affects what data is *returned*, not what data is *searched*. The IN clause affects what data is searched.

Example RETURNING Clauses

Search Type	Example(s)
No Field Spec	`Find {MyProspect}`
One sObject, no fields	`Find {MyProspect} RETURNING Contact`
Multiple sObjects, no fields	`Find {MyProspect} RETURNING Contact, Lead`
One sObject, one or more fields	`Find {MyProspect} RETURNING Contact(id)`
	`Find {MyProspect} RETURNING Contact(FirstName, LastName)`
	`Find {MyProspect} RETURNING Account(name, id)`
Custom sObject	`Find {MyProspect} RETURNING CustomObject_c`
	`Find {MyProspect} RETURNING CustomObject_c(id)`
	`Find {MyProspect} RETURNING CustomObject_c(id, CustomField_c)`
Multiple sObjects, one or more fields	`Find {MyProspect} RETURNING Contact(FirstName, LastName), Account(name, id)`
Multiple sObjects, mixed number of fields	`Find {MyProspect} RETURNING Contact(FirstName, LastName), Account, Lead(FirstName)`
Unsearchable sObjects	`Find {MyProspect} RETURNING RecordType(id)`
	`Find {MyProspect} RETURNING Pricebook`
Invalid sObjects	`Find {MyProspect} RETURNING FooBar`
Invalid sObject field	`Find {MyProspect} RETURNING Contact(fooBar)`

LIMIT n

The optional `LIMIT` clause allows you to specify the maximum number of rows returned in the text query. If unspecified, then the default is 200, which is the highest allowable value.

Example Text Searches

Look for `joe` anywhere in the system. Return the ids of the records where `joe` is found.

```
Find {joe}
```

Look for the name `Joe Smith` anywhere in the system, in a case insensitive way. Return the ids of the records where `Joe Smith` is found.

```
Find {Joe Smith}
```

Look for the name `Joe Smith` in the name field of a lead, return the id field of the records.

```
Find {Joe Smith}
In All Fields
Returning lead
```


Look for the name `Joe Smith` in the name field of a lead and return the name and phone number.

```
Find {Joe Smith}
In All Fields
Returning lead(name, phone)
```

Look for the name `Joe Smith` or `Joe Smythe` in the name field of a lead or contact and return the name and phone number. If an opportunity is called `Joe Smith`, the opportunity should not be returned.

```
Find {"Joe Smith" OR "Joe Smythe"}
In All Fields
Returning lead(name, phone), contact(name, phone)
```

Wildcards:

```
Find {Joe Sm*}
Find {Joe Sm?th*}
```

Delimiting "and" and "or" as literals when used alone:

```
Find {"and" or "or"}
Find {"joe and mary"}
Find {in}
Find {returning}
Find {find}
```

Escaping special characters & | ! ( ) { } [ ] ^ " ~ * ? : \ '

```
Find {right brace \}}
Find {asterisk \*}
Find {question \?}
Find {single quote  \'}
Find {double quote  \"}
```

Text Searches in Different Languages

In Chinese, Japanese, and Korean (CJK), words are delimited by pairs of CJK-type characters.

DATA REPLICATION

The sforce API supports *data replication*, which allows you to store and maintain a local, separate copy of your organization's pertinent salesforce.com data for specialized uses, such as data warehousing, data mining, custom reporting, analytics, integration with other applications, and so on. Data replication provides you with local control and the ability to run large or ad hoc analytical queries across the entire data set without transmitting all that data across the network.

API Calls for Data Replication

The sforce API supports data replication with the following API calls:

API Call	Description
getUpdated	Retrieves the list of objects that have been updated (added or changed) during the specified timespan for the specified object.
getDeleted	Retrieves the list of objects that have been deleted during the specified timespan for the specified object.

Client applications can invoke these API calls to determine which objects in your organization's data have been updated or deleted during a given time period. These API calls return a set of IDs for objects that have been updated (added or changed) or deleted, as well as the timestamp (GMT—not local—timezone) indicating when they were last updated or deleted. It is the responsibility of the client application to process these results and to incorporate the required changes into the local copy of the data.

Scope of Data Replication

This feature provides a mechanism that targets data replication (one-way copying of data). It does *not* provide data synchronization (two-way copying of data) or data mirroring capabilities.

Data Replication Steps

A client application typically proceeds along the following basic steps for *each* object that it replicates:

1. Optionally, the client application determines whether the structure of the object has changed since the last replication request, as described in Checking for Structural Changes in the Object on page 42.
2. Call getUpdated, passing in the object and timespan for which to retrieve data.

   Note that getUpdated retrieves the IDs for data to which the logged in user has access. Data that is outside of the user's sharing model is *not* returned. The sforce Web service returns the ID of every changed object that is visible to you, regardless of what change occurred in the object.
3. Iterate through the returned array of IDs. For each ID element in the array, call retrieve to obtain the latest information you want from the associated object. The client application must then take the appropriate action on the local data, such as inserting new rows or updating existing ones with the latest information.
4. Call getDeleted, passing in the object and timespan for which to retrieve data.

   Unlike getUpdated, getDeleted retrieves the IDs of *all* deleted objects for the given object throughout the organization, including data that is outside of the user's sharing model.
5. Iterate through the returned array of IDs. Your client application must then take the appropriate action on the local data to remove (or flag as deleted) the deleted objects. If your client application cannot match rows in the local data using the retrieved object ID, then it must call retrieve, passing in the ID, to obtain the information it needs to match the rows in the local data to delete.
6. Optionally, the client application saves the request timespans for future reference.

Object-Specific Requirements for Data Replication

The sforce API objects have the following requirements for data replication:

- The getUpdated call filters the results so that the client application receives IDs for only those created or updated objects to which the logged in user has access.
  - Your client application can replicate any objects to which it has sufficient permissions. For example, to replicate *all* data for your organization, your client application must be logged in with "View All Data" access rights to the specified object. For more information, see Factors that Affect Data Access on page 23.

  - The objects must be within your sharing rules.

  The getDeleted call, on the other hand, returns the IDs of all deleted objects without filtering.

- The object must be configured to be replicateable (`replicateable=True`). To determine whether a given object is replicateable, your application can invoke the describeSObject call on the object and inspect the `replicateable` property in the DescribeSObjectResult.

Polling for Changes

Client applications typically poll for changed data periodically. Polling involves the following considerations:

- The polling frequency depends on business requirements for how quickly changes in your organization's salesforce.com data need to be reflected in the local copy. Some client applications might poll just once a day to retrieve changes, while other client applications might poll every five minutes to achieve near real-time redundant mirroring of the salesforce.com data.
- The sforce Web service truncates the seconds portion of dateTime values. For example, if a client application submits a timespan between 12:30:15 (GMT) and 12:35:15 (GMT), then the sforce Web service retrieves information about items that have changed between 12:30:00 (GMT) and 12:35:00 (GMT), inclusive.

  Note: Development tools differ in the way that they handle time data. Some development tools report the local time, while others report only the GMT time. To determine how your development tool handles time values, refer to its documentation.
- We recommend polling no more frequently than every five minutes. The sforce Web service has built in controls to prevent errant applications from invoking the data replication API calls too frequently.
- Client applications should save the timespan used in previous data replication API calls so that the application knows the last time period for which data replication was successfully completed.
- To ensure data integrity on the local copy of the data, a client application needs to capture all of the relevant changes during polling—even if it requires processing data redundantly to ensure that there are no gaps. You client application can contain business logic to skip processing objects that have already been integrated into your local data.
- Gaps can also occur if the client application somehow fails to poll the data as expected (for example, due to a hardware crash or network connection failure). Your client application can contain business logic that determines the last successful replication and polls for the next consecutive timespan.
- If for any reason the local data is compromised, your client application might also provide business logic for rebuilding the local data from scratch.

Checking for Structural Changes in the Object

In the sforce API, data replication focuses on changes made to *rows* of objects. It does not determine whether changes have been made to the *structure* of objects (for example, fields added to—or removed from—a custom object). It is the responsibility of the client application to check whether the structure of a given object has changed since the last update. Before replicating data, client applications can call describeSObject on the object, and then compare the data returned in the DescribeSObjectResult with the data returned and saved from previous describeSObject invocations.

LIST OF SFORCE API CALLS

Table 3: Supported Calls in the sforce API

Task / Call	Description
convertLead	Converts a Lead into an Account, Contact, or (optionally) an Opportunity.
create	Adds one or more new individual objects to your organization's data.
delete	Deletes one or more individual objects from your organization's data.
describeGlobal	Retrieves a list of available objects for your organization's data.
describeLayout	Retrieves metadata about page layouts for the specified object type.
describeSObject	Retrieves metadata (field list and object properties) for the specified object type.
getDeleted	Retrieves the IDs of individual objects of the specified object that have been deleted since the specified time.
getServerTimestamp	Retrieves the current system timestamp (GMT) from the sforce Web service.
getUpdated	Retrieves the IDs of individual objects of the specified object that have been updated since the specified time.
getUserInfo	Retrieves personal information for the user associated with the current session.
login	Logs in to the sforce server and starts a client session.
query	Executes a query against the specified object and returns data that matches the specified criteria.
queryMore	Retrieves the next batch of objects from a query.
resetPassword	Changes a user's password to a server-generated value.
retrieve	Retrieves one or more objects based on the specified object IDs.
search	Executes a text search in your organization's data.
setPassword	Sets the specified user's password to the specified value.
update	Updates one or more existing objects in your organization's data.

convertLead

Converts a Lead into an Account, Contact, or (optionally) an Opportunity.

Syntax

```
LeadConvertResult[] = sfdc.convertLead(leadConverts LeadConvert[]);
```

Usage

Use convertLead to convert a Lead into an Account and Contact, as well as (optionally) an Opportunity. To convert a Lead, your client application must be logged in with "Convert Leads" permission and "Edit" permission on Leads, as well as sufficient permissions to create or update, as applicable, any associated Account, Contact, and Opportunity.

The convertLead call provides an easy way to convert the information in a qualified lead to a new or updated account, contact, and opportunity. Your organization can set its own guidelines for determining when a lead is qualified, but typically, a lead can be converted as soon as it becomes a real opportunity that you want to forecast.

If data is merged into existing account and contact objects, then only empty fields in the target object are overwritten—existing data (including IDs) are *not* overwritten. The only exception to this is if your client application sets **`overwriteLeadSource`** to True, in which case the **`LeadSource`** field in the target Contact object will be overwritten with the contents of the **`LeadSource`** field in the source Lead object.

Rules and Guidelines

When converting leads, consider the following rules and guidelines:

Field Mappings. The system automatically maps standard lead fields to standard account, contact, and opportunity fields. For custom lead fields, your administrator can specify how they map to custom account, contact, and opportunity fields.

Record Types. If the organization uses record types, the default record type of the new owner is assigned to records created during lead conversion.

Picklist Values. The system assigns the default picklist values for the account, contact, and opportunity when mapping any standard lead picklist fields that are blank. If your organization uses record types, blank values are replaced with the default picklist values of the new record owner.

Basic Steps for Converting Leads

Converting leads involves the following basic steps:

1. The client application determines the IDs of any lead(s) to be converted.
2. Optionally, the client application determines the IDs of any account(s) to merge the lead into. The client application can use SOSL or SOQL to search for accounts that match the lead name, as in the following example,

   ```
   select id, name from account where name='CompanyNameOfLeadBeingMerged'
   ```
3. Optionally, the client application determines the IDs of contact(s) to merge the lead into. The client application can use SOSL or SOQL to search for contacts that match the lead contact name, as in the following example.

   ```
   select id, name from contact where firstName='FirstName' and
   lastName='LastName' and accountId = '001…'
   ```
4. Optionally, the client application determines whether opportunities should be created from the leads.
5. The client application queries the LeadSource table to obtain all of the possible converted status options (select … from LeadStatus where IsConverted='1'), and then selects a value for the Converted Status.
6. The client application calls convertLead.
7. The client application iterates through the returned result(s) and examine each LeadConvertResult object to determine whether conversion succeeded for each lead.
8. Optionally (best practice), the client application creates tasks in which the **`WhoId`** is the **`ContactId`** and (if an opportunity is created), the **`WhatId`** is the **`OpportunityId`**.

Sample Code—Java

```
private Boolean convertLead(ID leadId, ID contactId,
      ID accountId, boolean overWriteLeadSource,
      boolean doNotCreateOpportunity, String opportunityName,
      String convertedStatus, boolean sendEmailToOwner)
{

   LeadConvert leadConvert = new LeadConvert();
   leadConvert.setLeadId(leadId);
   leadConvert.setContactId(contactId);
   leadConvert.setAccountId(accountId);
   leadConvert.setOverwriteLeadSource(overWriteLeadSource);
   leadConvert.setDoNotCreateOpportunity(doNotCreateOpportunity);
   leadConvert.setOpportunityName(opportunityName);
   leadConvert.setConvertedStatus(convertedStatus);
   leadConvert.setSendNotificationEmail(sendEmailToOwner);

   LeadConvertResult[] lcr = null;
   try
   {

      lcr = binding.convertLead(new LeadConvert[] {leadConvert});
      for (int i=0;i<lcr.length;i++)
         if (lcr[i].isSuccess()){
            System.out.println("Conversion succeeded.\n");
            LeadConvertResult result = lcr[i];
            System.out.println("The new contact id is: " +
result.getContactId().getValue());
         } else {
            System.out.println("The conversion failed because: " +
lcr[i].getErrors(0).getMessage());
        }
   }

   catch (UnexpectedErrorFault e) {
      System.out.println("Unexpected error encountered:\n\n" +
e.getExceptionMessage());
      return new Boolean.FALSE;
   } catch (RemoteException e) {
      System.out.println("Remote exception encountered:\n\n" + e.getMessage());
      return new Boolean.FALSE;
   }
   return new Boolean(true);
}
```

Sample Code—C#

```
private bool convertLead(string leadId, string contactId,
   string accountId, bool overWriteLeadSource,
   bool doNotCreateOpportunity, string opportunityName,
   string convertedStatus, bool sendEmailToOwner)
{

   sforce.LeadConvert leadConvert = new sforce.LeadConvert();
   leadConvert.leadId = leadId;
   leadConvert.contactId = contactId;
   leadConvert.accountId = accountId;
```



```
   leadConvert.overwriteLeadSource = overWriteLeadSource;
   leadConvert.doNotCreateOpportunity = doNotCreateOpportunity;
   leadConvert.opportunityName = opportunityName;
   leadConvert.convertedStatus = convertedStatus;
   leadConvert.sendNotificationEmail = sendEmailToOwner;

   sforce.LeadConvertResult[] lcr = null;
   try
   {

      lcr = binding.convertLead(new sforce.LeadConvert[] {leadConvert});
      for (int i=0;i<lcr.Length;i++)
         if (lcr[i].success)
        {
            Console.WriteLine("Conversion succeeded.\n");
            sforce.LeadConvertResult result = lcr[i];
            Console.WriteLine("The new contact id is: " + result.contactId);
         }
         else
         {
            Console.WriteLine("The conversion failed because: " +
lcr[i].errors[0].message);
         }
   }

   catch (Exception e)
   {
      Console.WriteLine("Unexpected error encountered:\n\n" + e.Message);
      return false;
   }

   return true;
}
```

Arguments

The convertLead call accepts an array of LeadConvert objects (100 maximum). A LeadConvert object contains the following properties.

Name	Type	Description
accountId	ID	ID of the Account into which the lead will be merged. Required only when updating an existing account. If no `accountID` is specified, then the sforce Web service creates a new Account. To create a new Account, the client application must be logged in with sufficient access rights. To merge a Lead into an existing Account, the client application must be logged in with read/write access to the specified Account. The account name and other existing data are *not* overwritten.
contactId	ID	ID of the Contact into which the lead will be merged (this contact must be associated with the specified **accountId**, and an **accountId** must be specified). Required only when updating an existing contact. If no `contactID` is specified, then the sforce Web service creates a new Contact that is implicitly associated with the Account. To create a new Contact, the client application must be logged in with sufficient access rights. To merge a Lead into an existing Contact, the client application must be logged in with read/write access to the specified Contact. The contact name and other existing data are *not* overwritten (unless **overwriteLeadSource** is set to True, in which case only the **LeadSource** field is overwritten).
convertedStatus	string	Valid LeadStatus value for a converted Lead. Required. To obtain the list of possible values, the client application queries the LeadStatus object, as in: `Select Id, MasterLabel from LeadStatus where IsConverted=true`
doNotCreateOpportunity	boolean	Specifies whether to create an Opportunity during lead conversion (False, the default) or not (True). Set this flag to True only if you do *not* want to create an Opportunity from the lead. An opportunity is created by default.
leadId	ID	ID of the Lead to convert. Required.
opportunityName	string	Name of the Opportunity to create. If no name is specified, then this value defaults to the company name of the lead. The maximum length of this field is 80 characters. If **doNotCreateOpportunity** argument is True, then no Opportunity is created and this field must be left blank; otherwise, an error is returned.

Name	Type	Description
overwriteLeadSource	boolean	Specifies whether to overwrite the **LeadSource** field on the target Contact object with the contents of the **LeadSource** field in the source Lead object (True), or not (False, the default). To set this field to True, the client application must specify a **contactId** for the target Contact object.
ownerId	ID	Specifies the ID of the person to own any newly created Account, Contact, and Opportunity. If the client application does not specify this value, then the owner of the new entities will be the owner of the Lead. Not applicable when merging with existing objects—if an **ownerId** is specified, the sforce Web service does not overwrite the **ownerId** field in an existing Account or Contact.
sendNotificationEmail	boolean	Specifies whether to send a notification email to the owner specified in the **ownerId** (True) or not (False, the default).

Response

LeadConvertResult[]

Fault

`UnexpectedErrorFault`

See Also

Sample SOAP Messages—convertLead
Concepts on page 22

LeadConvertResult

The convertLead call returns an array of `LeadConvertResult` objects. Each element in the `LeadConvertResult` array corresponds to the LeadConvert[] array passed as the `leadConverts` parameter in the convertLead call. For example, the object returned in the first index in the `LeadConvertResult` array matches the object specified in the first index of the LeadConvert[] array. A `LeadConvertResult` object has the following properties:

Name	Type	Description
accountId	ID	ID of the new Account (if a new account was specified) or the ID of the account specified when convertLead was invoked.
contactId	ID	ID of the new Contact (if a new contact was specified) or the ID of the contact specified when convertLead was invoked.
leadId	ID	ID of the converted Lead.
opportunityId	ID	ID of the new Opportunity, if one was created when convertLead was invoked.

Name	Type	Description
success	boolean	Indicates whether the convertLead call succeeded (True) or not (False) for this object.
errors	Error[]	If an error occurred during the create call, an array of one or more Error objects providing the error code and description.

create

Adds one or more new individual objects to your organization's data.

Syntax

```
SaveResult[] = sfdc.create(sObject[] sObjects);
```

Usage

Use create to add one or more individual objects, such as an Account or Contact, to your organization's information. The create call is analogous to the INSERT statement in SQL.

Rules and Guidelines

When creating objects, consider the following rules and guidelines:

Permissions. Your client application must be logged in with sufficient access rights to create individual objects within the specified object. For more information, see Factors that Affect Data Access on page 23.

Special Handling. Certain objects—and certain fields within those objects—require special handling or permissions. For example, you might also need permissions to access this object's parent object. Before you attempt to create a particular object, be sure to read its description in Chapter 5: sforce Objects on page 101.

Createable Fields. Certain objects cannot be created via the sforce API. To create an object via the create call, its object must be configured as createable (`createable=True`). To determine whether a given object can be created, your client application can invoke the describeSObject call on the object and inspect its `createable` property.

Automatically Maintained Fields. The sforce Web service generates unique values for ID fields automatically. For create, you cannot explicitly specify an ID value in the sObject. The SaveResult contains the ID of each object that was successfully created.

The sforce Web service populates certain fields automatically, such as **`CreatedDate`**, **`CreatedById`**, **`LastModifiedDate`**, **`LastModifiedById`**, and **`SystemModstamp`**. You cannot explicitly specify these values.

Required Fields. For required fields that do not have a preconfigured default value, you must supply a value. For more information, see Required Fields on page 107.

Default Values. For some objects, certain fields have a default value, such as **`OwnerID`**. If you do not specify a value for such fields, the sforce Web service populates these fields with the default value. For example, if you do not override the **`OwnerID`**, then the sforce Web service populates this field with the user ID associated with the user under which your client application is logged in.

- For required fields that do not have a preconfigured default value, you must supply a value.
- For all other fields in the object, if you do not explicitly specify a value, then its value is `null`.

Referential Integrity. Your client application must conform to the rules of referential integrity. For example, if you are creating an object that is the child of a parent object, you must supply the foreign key information that links the child to the parent. For example, when creating a CaseComment, you must supply the valid caseID for the parent Case, and that parent Case must exist in the database.

Valid Data 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 specified for your programming language and development tool (your development tool will handle the appropriate mapping of data types in SOAP messages).

String Values. When storing values in string fields, the sforce Web service trims any leading and trailing whitespace. For example, if the value of a name field is entered as `"  ABC Company "`, then the value is stored in the database as `"ABC Company"`.

Assignment Rules. When creating new Case or Lead objects, your client application can set options in the AssignmentRuleHeader to have the case or lead automatically assigned to one or more users based on assignment rules configured in the salesforce.com user interface. For more information, see Case on page 126 or Lead on page 141.

Maximum Number of Objects Created. Your client application can add up to 200 individual objects in a single create call. If a create request exceeds 200 objects, then the entire operation fails.

Basic Steps for Creating Objects

Creating objects involves the following basic steps:

1. Instantiate one or more individual objects within the object. For each object, you populate its fields with the data that you want to add.
2. Construct an `sObject[]` array and populate that array with the objects that you want to create. All objects *must* be of the same object.
3. Call create, passing in the `sObject[]` array.
4. Process the results in the `SaveResult[]` object to verify whether the objects have been successfully created.

Sample Code—Java

```
public void createAccountSample() {

   // Create two account objects
   Account account1 = new Account();
   Account account2 = new Account();

   // Set some fields on the account object
   // Name field (required) not being set on account1,
   // so this record should fail during create.

   account1.setAccountNumber("002DF99ELK9");
   account1.setBillingCity("Wichita");
   account1.setBillingCountry("US");
   account1.setBillingState("KA");
   account1.setBillingStreet("4322 Haystack Boulevard");
   account1.setBillingPostalCode("87901");

   // Set some fields on the account2 object
   account2.setName("Golden Straw");
   account2.setAccountNumber("003DF99ELK9");
   account2.setBillingCity("Oaklanc");
   account2.setBillingCountry("US");
```



```
   account2.setBillingState("CA");
   account2.setBillingStreet("666 Raiders Boulevard");
   account2.setBillingPostalCode("97502");

   // Create an array of SObjects to hold the accounts
   SObject[] sObjects = new SObject[2];

   // Add the accounts to the SObject array
   sObjects[0] = account1;
   sObjects[1] = account2;

   // Invoke the create call
   SaveResult[] saveResults = binding.create(sObjects);

   // Handle the results
   for (int i=0;i<saveResults.length;i++) {
      // Determine whether create succeeded or had errors
      if (saveResults[i].isSuccess()) {
         // No errors, so we will retrieve the id created for this index
         System.out.println(saveResults[i].getId().getValue());
      }
      else {
         // Handle the errors
         ...
      }
   }
}
```

Sample Code—C#

```
private void createAccount()
{
   // Create an account object to send to the service
   Account account = new Account();

   // Set several properties
   account.Name = "Koka Kola";
   account.Website = "www.kokakola.com";

   // Add the account to an array of SObjects
   sObject[] records = new sObject[] {account};

   // Invoke the create call, passing in the account properties
   // and saving the results in a SaveResult object
   SaveResult[] saveResults = binding.create(records);

   // Access the new ID
   String newID = saveResults[0].id;
}
```


Arguments

Name	Type	Description
`sObjects`	sObject[]	Array of one or more objects (up to 200) to create. The sforce Web service creates these objects in array index order.

Response

SaveResult[]

Fault

`InvalidSObjectFault`
`UnexpectedErrorFault`

See Also

Sample SOAP Messages—create
Concepts on page 22

SaveResult

The create call returns an array of `SaveResult` objects. Each element in the `SaveResult` array corresponds to the sObject[] array passed as the `sObjects` parameter in the create call. For example, the object returned in the first index in the `SaveResult` array matches the object specified in the first index of the sObject[] array. A `SaveResult` object has the following properties:

Name	Type	Description
`id`	ID	ID of the sObject that you attempted to create. If this field contains a value, then the object was created successfully. If this field is empty, then the object was not created and the sforce Web service returned error information instead.
`success`	boolean	Indicates whether the create call succeeded (True) or not (False) for this object.
`errors`	Error[]	If an error occurred during the create call, an array of one or more Error objects providing the error code and description.

delete

Deletes one or more individual objects from your organization's data.

Syntax

```
DeleteResult[] = sfdc.delete(ID[] ids);
```


Usage

Use delete to delete one or more existing objects, such as individual accounts or contacts, in your organization's data. The delete call is analogous to the DELETE statement in SQL.

Rules and Guidelines

When deleting objects, consider the following rules and guidelines:

- Your client application must be logged in with sufficient access rights to delete individual objects within the specified object. For more information, see Factors that Affect Data Access on page 23.
- In addition, you might also need permissions to access this object's parent object. For special access requirements, see the object's description in Chapter 5: sforce Objects on page 101.
- To ensure referential integrity, the delete call supports cascading deletions. If you delete a parent object, you delete its children automatically, as long as each child object can be deleted. For example, if you delete a Case, the sforce API automatically deletes any CaseComment, CaseHistory, and CaseSolution objects associated with that case. However, if a CaseComment is not deletable or is currently being used, then the delete call on the parent Case will fail.
- Certain objects cannot be deleted via the sforce API. To delete an object via the delete call, its object must be configured as deletable (`deletable=True`). To determine whether a given object can be deleted, your client application can invoke the describeSObject call on the object and inspect its `deletable` property.

Basic Steps for Deleting Objects

Deleting objects involves the following basic steps:

1. Determine the ID of each object that you want to delete. For example, you might call query to retrieve a set of records that you want to delete based on specific criteria.
2. Construct an `ID[]` array and populate it with the IDs of each object that you want to delete. You can specify the IDs of different objects. For example, you could specify the ID for an individual Account and an individual Contact in the same array.
3. Call delete, passing in the `ID[]` array.
4. Process the results in the `DeleteResult[]` object to verify whether the objects have been successfully deleted.

Sample Code—Java

```
public void deleteSample() {

   // Create an array of IDs to hold the IDs of the records to delete
   ID[] ids = new ID[2];

   // Add the IDs to the ID array
   ids[0].setValue("001x00000000JerAAE");
   ids[1].setValue("001x00000000JesAAE");

   // Invoke the delete call
   DeleteResult[] deleteResults = binding.delete(tasks);
   // Process the results
   for (int i=0;i<deleteResults.length;i++) {
      DeleteResult deleteResult = deleteResults[i];
      // Determine whether delete succeeded or had errors
      if (deleteResult.isSuccess()) {
         // Get the id of the deleted record
         deleteResult.getId();
         }
```



```
      else {
         // Handle the errors
         Error[] errors = deleteResult.getErrors();
      }
   }
}
```

Sample Code—C#

```
private void deleteAccount()
{
   // Delete call takes an string array of Ids as parameter
   String[] IDs = new Sring[] {""};

   // Invoke the delete call, saving the result in a DeleteResult object
   DeleteResult[] deleteResults = binding.delete(IDs);

   // Determine whether the delete call succeeded or failed
   if (deleteResults[0].success)
   {
      // Delete operation succeeded
      System.Diagnostics.Trace.WriteLine("Deleted: " + deleteResults[0].id);
   }
   else
   {
      // Delete operation failed
      System.Diagnostics.Trace.WriteLine("Couldn't delete because: " +
deleteResults[0].errors[0].message);
   }
}
```

Arguments

Name	Type	Description
ids	ID[]	Array of one or more IDs associated with the objects to delete. The sforce Web service deletes these objects in array index order. You can pass a maximum of 2000 object IDs to the delete call.

Response

DeleteResult[]

Fault

InvalidSObjectFault

UnexpectedErrorFault

See Also

Sample SOAP Messages—delete
Concepts *on page 22*

DeleteResult

The delete call returns an array of `DeleteResult` objects. Each element in the `DeleteResult` array corresponds to the ID[] array passed as the `ids` parameter in the delete call. For example, the object returned in the first index in the `DeleteResult` array matches the object specified in the first index of the ID[] array.

A `DeleteResult` object has the following properties:

Name	Type	Description
id	ID	ID of an sObject that you attempted to delete.
success	boolean	Indicates whether the delete call succeeded (True) or not (False) for this object.
errors	Error[]	If an error occurred during the delete call, an array of one or more Error objects providing the error information.

describeGlobal

Retrieves a list of available objects for your organization's data.

Syntax

```
DescribeGlobalResult = sfdc.describeGlobal();
```

Usage

Use describeGlobal to obtain the list of available objects for your organization. You can then iterate through this list and use describeSObject to obtain metadata about individual objects.

Your client application must be logged in with sufficient access rights to retrieve metadata about your organization's data. For more information, see Factors that Affect Data Access on page 23.

Sample Code—Java

```
public void describeGlobalSample() {

   // Invoke describeGlobal call and save results in DescribeGlobalResult object
   DescribeGlobalResult describeGlobalResult = binding.describeGlobal();
   if (! (describeGlobalResult == null)) {
      // Get the array of object names from the result
      String[] types = describeGlobalResult.getTypes();
      if (! (types == null)) {
         for (int i = 0; i < types.length; i++) {
            System.out.println((types[i]));
         }
      }
   }
}
```

Sample Code—C#

```
private void globalDescribe()
```



```
{
   // Invoke describeGlobal call and save results in DescribeGlobalResult object
   DescribeGlobalResult dgr = binding.describeGlobal();

   // Iterate through the results
   for (int i=0;i<dgr.types.Length;i++)
   {
      // The dgr.types[i] object is a string
      System.Diagnostics.Trace.WriteLine(dgr.types[i]);
   }
   binding.describeSObject
}
```

Arguments

None.

Response

DescribeGlobalResult

Fault

UnexpectedErrorFault

See Also

describeSObject on page 63
Sample SOAP Messages—describeGlobal
Concepts on page 22
sforce Partner Web Services API on page 186

DescribeGlobalResult

The describeGlobal call returns a `DescribeGlobalResult` object, which has the following properties.

Name	Type	Description
encoding	string	Specifies how an organization's data is encoded, such as `UTF-8` or `ISO8859/1`.
maxBatchSize	int	Maximum number of records allowed in a create, update, or delete call.
types	string[]	List of available objects for your organization. You iterate through this list to retrieve the object string that you pass to describeSObject.

Retrieves metadata about page layouts for the specified object type.

describeLayout

Syntax

```
DescribeLayoutResult = sfdc.describeLayout(string sObjectType);
```

Usage

Use describeLayout to retrieve information about the layout (presentation of data to users) for a given object type. The describeLayout call returns metadata about a given page layout, including layouts for edit and display-only views and record type mappings. Note that field-level security and layout editability affects which fields appear in a layout.

Users typically have one layout per object type assigned in their user profiles. In the Enterprise Edition, users can have multiple layouts, associated with multiple record types, defined in their user profile. The describeLayout call returns metadata for multiple layouts, if applicable.

Note: The describeLayout call is an advanced API call that is typically used only by partners who have written custom page rendering code for generating output on a specialized device (for example, on PDAs) and need to examine the layout details of an object before rendering the page output.

Basic Steps for Describing Layouts

Describing layouts involves the following basic steps:

1. To display a detail page, a client application first gets the recordTypeId from the record, then it finds the layoutId associated with that recordTypeId (through recordTypeMapping), and finally it uses that layout information to render the page.
2. To display an edit page, a client application first determines whether more than one record type is available and, if so, presents the user with a choice. Once a record type has been chosen, then the client application uses the layout information to render the page. It uses the picklist values from the RecordTypeMapping to display valid picklist values for picklist fields.
3. A client application uses describeSObject to get the labels for layout, using the `value` field on DescribeLayoutComponent to perform a lookup on the DescribeSObjectResult when the LayoutComponent is of type `field`.

Sample Code—Java

```
private void describeLayoutSample() {
   try {
   String objectToDescribe = getUserInput("Enter an object name: ");
   DescribeLayoutResult dlr = binding.describeLayout(objectToDescribe);
   System.out.println("There are " + dlr.getLayouts().length +
   " layouts for the " + objectToDescribe + " object.");
   for (int i=0;i<dlr.getLayouts().length;i++){
      DescribeLayout layout = dlr.getLayouts(i);
      System.out.println("    There are " +
      layout.getDetailLayoutSections().length + " detail layout sections");
      for (int j=0;j<layout.getDetailLayoutSections().length;j++){
         DescribeLayoutSection dls = layout.getDetailLayoutSections(j);
         System.out.println(new Integer(j).toString() +
         "        This section has a heading of " + dls.getHeading());

         if (layout.getEditLayoutSections() != null) {
            System.out.println("    There are " +
            layout.getEditLayoutSections().length + " edit layout sections");
            for (int j=0;j<layout.getEditLayoutSections().length;j++){
```

```
                DescribeLayoutSection els = layout.getEditLayoutSections(j);
                System.out.println(new Integer(j).toString() +
                "        This section has a heading of " + els.getHeading());
                System.out.println("This section has " +
                els.getLayoutRows().length + " layout rows.");
                for (int k=0;k<els.getLayoutRows().length;k++){
                   DescribeLayoutRow lr = els.getLayoutRows(k);
                   System.out.println("            This row has " +
                   lr.getNumItems() + " items.");
                   for (int h=0;h<lr.getLayoutItems().length;h++){
                      DescribeLayoutItem li = lr.getLayoutItems(h);
                      if (li.getLayoutComponents() != null)
                      System.out.println("                " +
                      new Integer(h).toString() +
                      " " + li.getLayoutComponents(0).getValue());
                   }
                }
             }
          }
          if (dlr.getRecordTypeMappings() != null)
             System.out.println("There are " +
              dlr.getRecordTypeMappings().length + " record type mappings for the " +
                objectToDescribe + " object");
          else
             System.out.println("There are no record type mappings for the " +
             objectToDescribe + " object.");

       } catch (Exception e) {
          System.out.println("An exceptions was caught: " + e.getMessage());
       }
    }
```

Sample Code—C#

```
private void describeLayoutSample()
{
   try
   {
      Console.Write("Enter the name of an object to describe the layout of: ");
      string objectToDescribe = Console.ReadLine();
      sforce.DescribeLayoutResult dlr = binding.describeLayout(objectToDescribe);
      Console.WriteLine("There are " + dlr.layouts.Length + " layouts for the " +
         objectToDescribe + " object.");
      for (int i=0;i<dlr.layouts.Length;i++)
      {
         sforce.DescribeLayout layout = dlr.layouts[i];
         Console.WriteLine("    There are " + layout.detailLayoutSections.Length +
         " detail layout sections");
         for (int j=0;j<layout.detailLayoutSections.Length;j++)
         {
            sforce.DescribeLayoutSection dls = layout.detailLayoutSections[j];
            Console.WriteLine(j + "        This section has a heading of " +
         dls.heading);
         }
         if (layout.editLayoutSections != null)
         {
            Console.WriteLine("    There are " + layout.editLayoutSections.Length
            + " edit layout sections");
```



```
                for (int j=0;j<layout.editLayoutSections.Length;j++)
                {
                   sforce.DescribeLayoutSection els = layout.editLayoutSections[j];
                   Console.WriteLine(j + "        This section has a heading of " +
                   els.heading);
                   Console.WriteLine("This section has " + els.layoutRows.Length +
                   " layout rows.");
                   for (int k=0;k<els.layoutRows.Length;k++)
                   {
                      sforce.DescribeLayoutRow lr = els.layoutRows[k];
                      Console.WriteLine("            This row has " + lr.numItems +
                      " items.");
                      for (int h=0;h<lr.layoutItems.Length;h++)
                      {
                         sforce.DescribeLayoutItem li = lr.layoutItems[h];
                         if (li.layoutComponents != null)
                            Console.WriteLine("                " + h + " " +
                               li.layoutComponents[0].value);
                      }
                   }
                }
             }
          }
          if (dlr.recordTypeMappings != null)
             Console.WriteLine("There are " + dlr.recordTypeMappings.Length +
             " record type mappings for the " + objectToDescribe + " object");
          else
             Console.WriteLine("There are no record type mappings for the " +
             objectToDescribe + " object.");
       }
       catch (Exception e)
       {
          Console.WriteLine("An exceptions was caught: " + e.Message);
       }
    }
```

Arguments

Name	Type	Description
sObjectType	string	The specified value must be a valid object for your organization. For a complete list of sforce objects, see List of sforce Objects.

Response

DescribeLayoutResult

Fault

InvalidSObjectFault

UnexpectedErrorFault

See Also

Sample SOAP Messages—describeLayout
Concepts on page 22

DescribeLayoutResult

The describeLayout call returns a `DescribeLayoutResult` object containing top-level record type information about the passed-in **`sObjectType`**, as well as a mapping of record types to layouts. Your client application can traverse this object to retrieve detailed metadata about the layout.

Name	Type	Description
`layouts`	DescribeLayout[]	Layout(s) associated with the specified **`sObjectType`**. A user profile typically has only one associated layout per object type. However, a user profile might have multiple layouts associated with multiple record types.
`recordTypeMappings`	RecordTypeMapping[ ]	Record type mapping(s) available for the user. A user profile may have multiple record types. *All* record types are returned—not just those available to the calling user. This allows the client application to display a layout appropriate for a given user profile. For example, suppose User A owns a record, and this record has recordType X set. If User B tries to view this record, then the client application can display the record using the layout associated with this recordType for User B's profile (even if the record type is not a available for the user).

DescribeLayout

Represents a specific layout for the specified **`sObjectType`**. Each DescribeLayout is referenced by its unique layout ID and consists of two types of views (represented in this object as arrays of DescribeLayoutSections):

- **Detail view**—Read-only display of the object. In a detail layout, certain pieces of information (such as address details) might be aggregated into a single DescribeLayoutItem.
- **Edit view**—Editable display of the object. In an edit layout, individual pieces of information (such as an address) will be broken up into separate fields.

An individual DescribeLayout consists of the following fields:

Name	Type	Description
`detailLayoutSections`	DescribeLayoutSection[]	Layout section(s) for the detail view.
`editLayoutSections`	DescribeLayoutSection[]	Layout section(s) for the edit view.
`id`	ID	Unique ID of this Layout.

DescribeLayoutSection

Represents a section of a DescribeLayout and consists of one or more columns and one or more rows (an array of DescribeLayoutRows).

Name	Type	Description
`columns`	int	Number of columns in this DescribeLayoutSection.
`heading`	string	Heading text (label) for this DescribeLayoutSection.
`layoutRows`	DescribeLayoutRow[]	Array of one or more DescribeLayoutRows.
`rows`	int	Number of rows in this DescribeLayoutSection.
`useHeading`	boolean	Indicates whether to use the `heading` (True) or not (False).

DescribeLayoutRow

Represents a row in a DescribeLayoutSection. A DescribeLayoutRow consists of one or more DescribeLayoutItems. For each DescribeLayoutRow, a DescribeLayoutItem refers either to a specific field or to an "empty" DescribeLayoutItem (a DescribeLayoutItem that contains no DescribeLayoutComponents). An empty DescribeLayoutItem can be returned when a given DescribeLayoutRow is sparse (for example, containing more fields on the right column than on the left column). Where there are gaps in the layout, an empty DescribeLayoutItem is returned as a placeholder.

Name	Type	Description
`layoutItems`	DescribeLayoutItem[]	Refers to either a specific field or to an "empty" LayoutItem (a LayoutItem that contains no DescribeLayoutComponents).
`numItems`	int	Number of `layoutItems`. This information is redundant but, due to a bug in a popular SOAP toolkit, was required to avoid serialization problems.

DescribeLayoutItem

Represents an individual item in a DescribeLayoutRow. A DescribeLayoutItem consists of a set of components (DescribeLayoutComponent), each of which is either a field or a separator. For most fields on a layout, there is only one component per layout item. However, in a display-only view, the DescribeLayoutItem might be a composite of the individual fields (for example, an address can consist of street, city, state, country, and postal code data). On the corresponding edit view, each component of the address field would be split up into separate DescribeLayoutItems.

Name	Type	Description
editable	boolean	Indicates whether this DescribeLayoutItem can be edited (True) or not (False).
label	string	Label text for this DescribeLayoutItem.
layoutComponents	DescribeLayoutComponent[]	DescribeLayoutComponent(s) for this DescribeLayoutItem.
placeholder	boolean	Indicates whether this DescribeLayoutItem is a placeholder (True) or not (False). If True, then this DescribeLayoutItem is blank.
required	boolean	Indicates whether this DescribeLayoutItem is required (True) or not (False). This is useful to know if, for example, you wanted to render required fields in a contrasting color (such as red).

DescribeLayoutComponent

Represents the smallest unit in a layout—a field or a separator. To reference a field for display, a client application uses the following notation to reference a field in the describeSObject call: `LayoutComponent.`*`fieldName`*.

Name	Type	Description
tabOrder	int	Indicates the tab order for the item in the row.
type	layoutComponentType	Type of DescribeLayoutComponent. One of the following values: • `Field`—Field name. A mapping to the `name` field on the DescribeSObjectResult. • `Separator`—Separator character, such as a semicolon (:) or slash (/).
value	string	Value of this DescribeLayoutComponent.

RecordTypeMapping

Represents a single record type mapping in the `recordTypeMappings` field in a DescribeLayoutResult object. This object is a map of valid `recordTypeId`s to `layoutId`s. For displaying a detail view, a client application uses this mapping to determine which layout is associated with the record type on the record. For displaying an edit view, a client application uses this mapping to determine which layout to use (and possibly to allow the user to choose between multiple record types); it will also determine the set of available picklist values.

Name	Type	Description
available	boolean	Indicates whether this record type mapping is available (True) or not (False). Availability is used to display a list of available record types to the user when they are creating a new record.
defaultRecordTypeMapping	boolean	Indicates whether this is the default record type mapping (True) or not (False).
layoutId	ID	ID of the layout associated with this record type mapping.
name	string	Name of this record type mapping.
recordTypeId	ID	ID of the record type to map.
picklistsForRecordType	PicklistForRecordType[]	Record type picklist(s) mapped to the **recordTypeId**.

PicklistForRecordType

Represents a single record type picklist in a RecordTypeMapping. The `picklistName` matches up with the `name` field in the DescribeSObjectResult, and the PicklistForRecordTypes are the set of acceptable values for the `recordType`.

Name	Type	Description
picklistName	string	Name of the picklist.
picklistValues	PickListEntry[]	Set of picklist values associated with the **recordTypeId** in the RecordTypeMapping.

describeSObject

Describes metadata (field list and object properties) for the specified object.

Syntax

```
DescribeSObjectResult = sfdc.describeSObject(string sObjectType);
```

Usage

Use describeSObject to obtain metadata for a given object. You can first call describeGlobal to retrieve a list of all objects for your organization, then iterate through this list and use describeSObject to obtain metadata about individual objects.

Your client application must be logged in with sufficient access rights to retrieve metadata about your organization's data. For more information, see Factors that Affect Data Access on page 23.

Sample Code—Java

```
public void describeSample() {

   // Invoke describeSObject and save results in DescribeSObjectResult
   DescribeSObjectResult describeSObjectResult =
binding.describeSObject("account");
   // Determine whether the describeSObject call succeeded
   if (! (describeSObjectResult == null)) {
      // Retrieve fields from the results
      Field[] fields = describeSObjectResult.getFields();
      // Get the name of the object
      String objectName = describeSObjectResult.getName();
      // Get some flags
      boolean isActivateable = describeSObjectResult.isActivateable();
      // Many other values are accessible

      if (! (fields == null)) {
      // Iterate through the fields to get properties for each field
         for (int i = 0; i < fields.length; i++) {
            Field field = fields[i];
            int byteLength = field.getByteLength().intValue();
            int digits = field.getDigits().intValue();
            String label = field.getLabel();
            int length = field.getLength().intValue();
            String name = field.getName();
            PicklistEntry[] picklistValues = field.getPicklistValues();
            int precision = field.getPrecision().intValue();
            String[] referenceTos = field.getReferenceTo();
            int scale = field.getScale().intValue();
            FieldType fieldType = field.getType();
            boolean fieldIsCreateable = field.isCreateable();
            // Determine whether there are picklist values
            if (picklistValues != null) {
               System.out.println("Picklist values = ");
               for (int j = 0; j < picklistValues.length; j++) {
                  if (picklistValues[j].getLabel() != null) {
                     System.out.println("    Item:  " +
picklistValues[j].getLabel());
                  }
               }
            }
            // Determine whether this field refers to another object
            if (referenceTos != null) {
               System.out.println("Field references the following objects:");
               for (int j = 0; j < referenceTos.length; j++) {
                  System.out.println("    " + referenceTos[j]);
               }
            }
         }
      }
   }
}
```

Sample Code—C#

```
private void sObjectDescribe()
{
```



```
    // Invoke describeSObject and save results in DescribeSObjectResult
    DescribeSObjectResult dsr = binding.describeSObject("Account");

    // Get value that indicates whether we can create a record
    bool canCreate = dsr.createable;

    // Get a field and save its name
    String fldName = dsr.fields[0].name;
}
```

Arguments

Name	Type	Description
sObjectType	string	Object. The specified value must be a valid object for your organization. For a complete list of sforce objects, see List of sforce Objects.

Response

DescribeSObjectResult

Fault

InvalidSObjectFault

UnexpectedErrorFault

See Also

describeGlobal on page 55
Sample SOAP Messages—describeSObject
Concepts on page 22
sforce Partner Web Services API on page 186

DescribeSObjectResult

The describeSObject call returns a `DescribeSObjectResult` object, which has the following properties. Note that, while the boolean properties indicate whether certain API calls can be used for an object, other factors (such as security settings in the user's personal profile) also affect whether such operations can be performed on the object.

Name	Type	Description
activateable	boolean	Reserved for future use.
createable	boolean	Indicates whether the object can be created via the create call (True) or not (False).
custom	boolean	Indicates whether the object is a custom object (True) or not (False).

Name	Type	Description
deletable	boolean	Indicates whether the object can be deleted via the delete call (True) or not (False).
fields	Field[]	Array of fields associated with the object. The mechanism for retrieving information from this list varies among development tools.
keyPrefix	string	Three character code prefix in the object ID. Object IDs are prefixed with three character codes that specify the type of the object (for example, Account objects have a prefix of "001" and Opportunity objects have a prefix of "006". The describeSObject call returns a value for objects that have a stable prefix. For objects types that do not have a stable or predictable prefix, this field is blank. Therefore, client applications that rely on these codes should use this way of determining object types to ensure forward compatibility.
label	string	Label text for a renamed tab (for example, "Patient" in a medical vertical) in the user interface, if applicable, or the object name, if not.
layoutable	boolean	Indicates whether the object supports the describeLayout call (True) or not (False).
name	string	Name of the object. This is the same string that was passed in as the sObjectType parameter.
queryable	boolean	Indicates whether the object can be queried via the query call (True) or not (False).
replicateable	boolean	Indicates whether the object can be replicated via the getUpdated and getDeleted calls (True) or not (False).
retrieveable	boolean	Indicates whether the object can be retrieved via the retrieve call (True) or not (False).
searchable	boolean	Indicates whether the object can be searched via the search call (True) or not (False).
undeletable	boolean	Reserved for future use.
updateable	boolean	Indicates whether the object can be updated via the update call (True) or not (False).
urlDetail	string	URL to the view details screen (read-only) for this object. Compare with urlEdit, which is read-write. Client applications can use this URL to redirect to, or access, the standard salesforce.com user interface for standard and custom objects. To provide flexibility and allow for future enhancements, returned urlDetail values are dynamic. To ensure that client applications are forward compatible, it is recommended that they use this capability where possible. Note that, for objects for which a stable URL API is not available, this field is returned empty.

Name	Type	Description
urlEdit	string	URL to the edit screen for this object. For example, the **urlEdit** field for the Account object returns `https://na1.salesforce.com/{ID}/e`. Substituting the {*ID*} field for the current object ID will return the salesforce.com Web interface's edit page for that object specific account. Compare with `urlDetail`, which is read-only. Client applications can use this URL to redirect to, or access, the standard salesforce.com user interface for standard and custom objects. To provide flexibility and allow for future enhancements, returned `urlDetail` values are dynamic. To ensure that client applications are forward compatible, it is recommended that they use this capability where possible. Note that, for objects for which a stable URL API is not available, this field is returned empty.
urlNew	string	URL to the new/create screen for this object type. Client applications can use this URL to redirect to, or access, the standard salesforce.com user interface for standard and custom objects. To provide flexibility and allow for future enhancements, returned `urlNew` values are dynamic. To ensure that client applications are forward compatible, it is recommended that they use this capability where possible. Note that, for objects for which a stable URL API is not available, this field is returned empty.

Field

In the DescribeSObjectResult, the `fields` property contains an array of `Field` objects. Each field represents a field in an sforce API object. The array contains only the fields that the user can view, as defined by the user's field-level security settings.

Name	Type	Description
autonumber	boolean	Indicates whether this field is an autonumber field (`true`) or not (`false`). Analogous to a SQL `IDENTITY` type, autonumber fields are read only, non-createable text fields with a maximum length of 30 characters. Autonumber fields are read-only fields used to provide a unique ID that is independent of the internal object ID (such as a purchase order number or invoice number). Autonumber fields are configured *entirely* in the salesforce.com user interface. The sforce API provides access to this attribute so that client applications can determine whether a given field is an autonumber field.
byteLength	int	For variable-length fields (including binary fields), the maximum size of the field, in bytes.

Name	Type	Description
createable	boolean	Indicates whether the field can be created (True) or not (False). If True, then this field value can be set in a create call.
custom	boolean	Indicates whether the field is a custom field (True) or not (False).
defaultedOnCreate	boolean	Indicates whether this field is defaulted when created (`true`) or not (`false`). If True, then salesforce.com implicitly assigns a value for this field when the object is created, even if a value for this field is not passed in on the create call. For example, in the Opportunity object, the **Probability** field has this attribute because its value is derived from the **Stage** field. Similarly, the **Owner** has this attribute on most objects because its value is derived from the current user (if the **Owner** field is not specified).
digits	int	For fields of type integer. Maximum number of digits. The sforce Web service returns an error if an integer value exceeds the number of digits.
filterable	boolean	Indicates whether the field is filterable (True) or not (False). If True, then this field can be specified in the WHERE clause of a query string in a query call.
label	string	Text label that is displayed next to the field in the salesforce.com user interface. This label can be localized.
length	int	For string fields, the maximum size of the field in Unicode characters (not bytes).
name	string	Field name used in sforce API calls, such as create, delete, and query.
nameField	boolean	Indicates whether this field is a name field (True) or not (False). Used to identify the name field for standard objects (such as `AccountName` for an Account object) and custom objects. Limited to one per entity, except where `FirstName` and `LastName` fields are used (such as in the Contact object).
nillable	boolean	Indicates whether the field is nillable (True) or not (False). A nillable field can have empty content. A non-nillable field must have a value in order for the object to be created or saved.
picklistValues	PickListEntry[]	Provides the list of valid values for the picklist. Specified only if `restrictedPicklist` is True.
precision	int	For fields of type double. Maximum number of digits that can be stored, including all numbers to the left and to the right of the decimal point (but excluding the decimal point character).

Name	Type	Description
referenceTo	string[]	For fields that refer to other objects, this array indicates the objects of the referenced objects.
restrictedPicklist	boolean	Indicates whether the field is a restricted pick list (True) or not (False).
scale	int	For fields of type double. Number of digits to the right of the decimal point. The sforce Web service silently truncates any extra digits to the right of the decimal point, but it returns a fault response if the number has too many digits to the left of the decimal point.
selectable	boolean	Indicates whether the field is selectable (True) or not (False). If True, then this field can be specified in the list of fields of a query string in a query call.
soapType	SOAPType	See SOAPType for a list of allowable values.
type	FieldType	See FieldType for a list of allowable values.
updateable	boolean	Indicates whether the field is updateable (True) or not (False). If True, then this field value can be set in a update call.

FieldType

In the Field object associated with the DescribeSObjectResult, the `type` field can contain one of the following strings. For more information about field types, see Field Types on page 101.

Field Type	What the Field Contains
string	String values.
boolean	Boolean (True / False) values.
i4	Integer (int) values.
double	Double values.
date	Date values.
datetime	Date and time values.
base64	Base64-encoded arbitrary binary data (of type base64Binary). Used for Attachment, Document, and Scontrol objects.
id	Primary key field for the object.
reference	Cross-references to a different sforce object. Analogous to a foreign key field in SQL.
currency	Currency values.
textarea	String that is displayed as a multi-line text field.
percent	Percentage values.
phone	Phone numbers. Values can include alphabetic characters. Client applications are responsible for phone number formatting.

Field Type	What the Field Contains
`url`	URL values. Client applications should commonly display these as hyperlinks.
`email`	Email addresses.
`combobox`	Comboboxes, which provide a set of enumerated values and allow the user to specify a value not in the list.
`picklist`	Single-select picklists, which provide a set of enumerated values from which *only one* value can be selected.
`multipicklist`	Multi-select picklists, which provide a set of enumerated values from which *multiple* values can be selected.

SOAPType

In the Field property associated with the DescribeSObjectResult, the `soapType` field can contain any one of the following string values. All of the values preceded by **`xsd:`** are XML schema primitive data types. For more information about the XML schema primitive data types, see the World Wide Web Consortium's publication *XML Schema Part 2: Datatypes* at the following URL: http://www.w3.org/TR/xmlschema-2/.

Value	Description
`tns:ID`	Unique ID associated with an sObject.
`xsd:base64Binary`	Base 64-encoded binary data.
`xsd:boolean`	Boolean (True / False) values.
`xsd:dateTime`	Date/time values.
`xsd:double`	Double values.
`xsd:int`	Integer values.
`xsd:string`	Character strings.

PickListEntry

In the Field object associated with the DescribeSObjectResult, the `picklistValues` field contains an array of `PickListEntry` properties. Each `PickListEntry` can contain any one of the following string values. For more information, see Picklist Field Type on page 105.

Name	Type	Description
`label`	string	Display name of this item in the picklist.
`value`	string	Value of this item in the picklist.
`defaultValue`	boolean	Indicates whether this item is the default item (True) in the picklist or not (False). Only one item in a picklist is designated as the default.
`active`	boolean	Indicates whether this item must be displayed (True) or not (False) in the drop-down list for the picklist field in the user interface.

getDeleted

Retrieves the list of individual objects that have been deleted within the given timespan for the specified object.

Syntax

```
GetDeletedResult = sfdc.getDeleted(string sObjectType dateTime startDate dateTime
EndDate);
```

Usage

Use getDeleted for data replication applications to retrieve a list of object instances that have been deleted from your organization's data within the specified timespan. The getDeleted call retrieves a GetDeletedResult object that contains an array of `DeletedRecord` objects containing the ID of each deleted object and the date/time (GMT timezone) on which it was deleted. Be sure to read Data Replication on page 40 before using getDeleted in your client applications.

Note The getDeleted call retrieves the IDs of *all* deleted objects for the given object throughout the organization, including data that is outside of the user's sharing model.

Rules and Guidelines

When replicating deleted objects, consider the following rules and guidelines:

- The specified **`startDate`** must chronologically *precede* the specified **`endDate`** value. The specified **`startDate`** cannot be the same value as, or later than, the specified **`endDate`** value. Otherwise, the sforce Web service returns an `INVALID_REPLICATION_DATE` error.
- Client applications typically poll for changed data periodically. For important polling considerations, see Polling for Changes on page 42.
- Certain objects cannot be replicated via the sforce API. To replicate an object via the getDeleted call, its object must be configured as replicateable (`replicateable=True`). To determine whether a given object can be replicated, your client application can invoke the describeSObject call on the object and inspect its `replicateable` property.
- Development tools differ in the way that they handle time data. Some development tools report the local time, while others report only the GMT time. To determine how your development tool handles time values, refer to its documentation.

Basic Steps for Replicating Deleted Objects

Replicating objects involves the following basic steps for *each* object that you want to replicate:

1. Optionally, the client application determines whether the structure of the object has changed since the last replication request, as described in Checking for Structural Changes in the Object on page 42.
2. Call getDeleted, passing in the object and timespan for which to retrieve data.
3. In the GetDeletedResult object, iterate through the returned array of `DeletedRecord` objects containing the ID of each deleted object and the date/time (GMT timezone) on which it was deleted.
4. Your client application must then take the appropriate action on the local data to remove (or flag as deleted) the deleted objects. If your client application cannot match rows in the local data using the retrieved object ID, then it must call retrieve, passing in the ID, to obtain the information it needs to match the rows in the local data to delete.
5. Optionally, the client application saves the request timespan for future reference.

A client application likely performs other tasks associated with data replication operations. For example, if an opportunity were to become closed, a client application might run a new revenue

report. Similarly, if a task were completed, the process might log this somehow in another system.

Sample Code—Java

```
private void getDeletedSample() {
    //You can use the timestamp from the service for a known point in time
    Calendar serverTime = binding.getServerTimestamp().getTimestamp();
    //Create a start time value for the call
    GregorianCalendar startTime = (GregorianCalendar) serverTime.clone();
    //Create an end time value for the call
    GregorianCalendar endTime = (GregorianCalendar) serverTime;
    //Subtract 5 mins from the server time so that we have a valid time frame.
    //You can use just about any timespan you want, 5 minutes is arbitrary
    endTime.add(GregorianCalendar.MINUTE, -5);

    GetDeletedResult gdr = binding.getDeleted("Contact", (Calendar)startTime,
(Calendar)endTime);
    //Check the number of records contained in the results, if more that 0,
    //then something was deleted in the 5 minute span
    if (gdr.getDeletedRecords().length > 0) {
        for (int i=0;i<gdr.getDeletedRecords().length;i++) {
           System.out.println(gdr.getDeletedRecords(i).getId().getValue() + " was
deleted on " + gdr.getDeletedRecords(i).getDeletedDate().getTime().toString());
        }
    } else {
        System.out.println("No deletions from contacts in the last 5 minutes.");
    }
}
```

Sample Code—C#

```
private void getDeletedSample()
{
    DateTime endTime = binding.getServerTimestamp().timestamp;
    DateTime startTime = endTime.Subtract(new System.TimeSpan(0, 0, 5, 0, 0));
    sforce.GetDeletedResult gdr = binding.getDeleted("Contact", startTime,
endTime);

    if (gdr.deletedRecords.Length > 0)
    {
        for (int i=0;i<gdr.deletedRecords.Length;i++)
        {
            Console.WriteLine(gdr.deletedRecords[i].id + " was deleted on " +
gdr.deletedRecords[i].deletedDate.ToString());
        }
    }
    else
    {
       Console.WriteLine("No deleted contacts between " + startTime.ToString() +
" and " + endTime.ToString() );
    }
}
```


Arguments

Name	Type	Description
sObjectType	string	Object. The specified value must be a valid object for your organization. For a complete list of sforce objects, see List of sforce Objects.
startDate	dateTime	Starting date/time (GMT—not local—timezone) of the timespan for which to retrieve the data. The sforce Web service ignores the seconds portion of the specified dateTime value (for example, 12:30:15 is interpreted as 12:30:00 GMT).
endDate	dateTime	Ending date/time (GMT—not local—timezone) of the timespan for which to retrieve the data. The sforce Web service ignores the seconds portion of the specified dateTime value (for example, 12:35:15 is interpreted as 12:35:00 GMT).

Response

`GetDeletedResult`

Faults

`InvalidSObjectFault`

`UnexpectedErrorFault`

See Also

getUpdated
Data Replication on page 40
Sample SOAP Messages—getDeleted
Concepts on page 22

GetDeletedResult

The getDeleted call returns a `GetDeletedResult` object that contains an array of `DeletedRecord` objects. Each element in the `DeletedRecord` array corresponds to an object that was deleted within the given timespan. Each `DeletedRecord` object has the following properties:

Name	Type	Description
id	ID	ID of an sObject that has been deleted.
modifiedDate	dateTime	Date/time (GMT—not local—timezone) on which this object was deleted.

getUpdated

Retrieves the list of individual objects that have been updated (added or changed) within the given timespan for the specified object.

Syntax

```
GetUpdatedResult[] = sfdc.getUpdated(string sObjectType dateTime startDate
dateTime EndDate);
```

Usage

Use getUpdated for data replication applications to retrieve a set of IDs for objects of the specified object that have been created or updated within the specified timespan. The getUpdated call retrieves an array of `GetUpdatedResult` objects containing the ID of each created or updated object and the date/time (GMT timezone) on which it was created or updated, respectively. Be sure to read Data Replication on page 40 before using getUpdated in your client application.

Note

The getUpdated call retrieves the IDs only for objects to which the logged in user has access. For example, data that is outside of the user's sharing model is *not* returned.

Rules and Guidelines

When replicating created and updated objects, consider the following rules and guidelines:

- The specified **`startDate`** must chronologically *precede* the specified **`endDate`** value. The specified **`startDate`** cannot be the same value as, or later than, the specified **`endDate`** value. Otherwise, the sforce Web service returns an `INVALID_REPLICATION_DATE` error.
- Client applications typically poll for changed data periodically. For important polling considerations, see Polling for Changes on page 42.
- Your client application can replicate any objects to which it has sufficient permissions. For example, to replicate *all* data for your organization, your client application must be logged in with "View All Data" access rights to the specified object. Similarly, the objects must be within your sharing rules. For more information, see Factors that Affect Data Access on page 23.
- Certain objects cannot be replicated via the sforce API. To replicate an object via the getUpdated call, its object must be configured as replicateable (`replicateable=True`). To determine whether a given object can be replicated, your client application can invoke the describeSObject call on the object and inspect its `replicateable` property.
- Certain objects cannot be deleted, such as Group, User, Contract, or Product2 objects. However, if instances of these objects are no longer visible in the salesforce.com user interface, then they might have been rendered inactive so that only users with administrative access can see them. To determine whether a missing object instance has been made inactive, your client application can call getUpdated and check the object's active flag.
- Development tools differ in the way that they handle time data. Some development tools report the local time, while others report only the GMT time. To determine how your development tool handles time values, refer to its documentation.

Basic Steps for Replicating Updated Objects

Replicating objects involves the following basic steps for *each* object that you want to replicate:

1. Optionally, the client application determines whether the structure of the object has changed since the last replication request, as described in Checking for Structural Changes in the Object on page 42.
2. Call getUpdated, passing in the object and timespan for which to retrieve data.
3. Iterate through the returned array of IDs. For each ID element in the array, call retrieve to obtain the latest information you want from the associated object. Your client application must then take the appropriate action on the local data, such as inserting new rows or updating existing ones with the latest information.
4. Optionally, the client application saves the request timestamp for future reference.

A client application likely performs other tasks associated with data replication operations. For example, if an opportunity were to become closed, a client application might run a new revenue report. Similarly, if a task were completed, the process might log this somehow in another system.

Sample Code—Java

```
private void getUpdatedSample() {
    //You can use the server timestamp as known point in time
    Calendar serverTime = binding.getServerTimestamp().getTimestamp();
    //Create a start time value for the call
    GregorianCalendar startTime = (GregorianCalendar) serverTime.clone();
    //Create an end time value for the call
    GregorianCalendar endTime = (GregorianCalendar) serverTime;
    //subtract 5 mins from the server time so
    //that we have a valid time frame, you can use just
    //about any timespan you want, 5 minutes is arbitrary
    startTime.add(GregorianCalendar.MINUTE, -5);

    System.out.println("Checking updates at: " + startTime.getTime().toString());
    GetUpdatedResult ur = binding.getUpdated("Account",
                (Calendar)startTime, (Calendar)endTime);

    //Check the length of the returned array of IDs
    //to detect if you got any hits
    if (ur.getIds().length > 0) {
        for (int i=0;i<ur.getIds().length;i++) {
           System.out.println(ur.getIds(i).getValue() + " was updated between " +
startTime.getTime().toString() + " and " + endTime.getTime().toString());
        }
    } else {
        System.out.println("No updates to accounts in the last 5 minutes.");
    }
}
```

Sample Code—C#

```
private void getUpdatedSample()
{
    DateTime endTime = binding.getServerTimestamp().timestamp;
    DateTime startTime = endTime.Subtract(new System.TimeSpan(0, 0, 5, 0, 0));
    sforce.GetUpdatedResult gur = binding.getUpdated("Account", startTime,
endTime);
    if (gur.ids.Length > 0)
    {
        for (int i=0;i<gur.ids.Length;i++)
        {
            Console.WriteLine(gur.ids[i] + " was updated between " +
startTime.ToString() + " and " + endTime.ToString());
        }
    }
    else
    {
       Console.WriteLine("No updates to accounts between " + startTime.ToString()
+ " and " + endTime.ToString() );
    }
}
```


Arguments

Name	Type	Description
sObjectType	string	Object. The specified value must be a valid object for your organization. For a complete list of sforce objects, see List of sforce Objects.
startDate	dateTime	Starting date/time (GMT—not local—timezone) of the timespan for which to retrieve the data. The sforce Web service ignores the seconds portion of the specified dateTime value (for example, 12:30:15 is interpreted as 12:30:00 GMT).
endDate	dateTime	Ending date/time (GMT—not local—timezone) of the timespan for which to retrieve the data. The sforce Web service ignores the seconds portion of the specified dateTime value (for example, 12:35:15 is interpreted as 12:35:00 GMT).

Response

`GetUpdatedResult[]`

Fault

`InvalidSObjectFault`

`UnexpectedErrorFault`

See Also

getDeleted
Data Replication *on page 40*
Sample SOAP Messages—getDeleted
Concepts *on page 22*

GetUpdatedResult

The getUpdated call returns an array of `GetUpdatedResult` objects. Each element in the `GetUpdatedResult` array corresponds to an object that was inserted or updated within the given timespan. A `GetUpdatedResult` object has the following properties:

Name	Type	Description
id	ID	ID of an sObject that has been updated.

login

Logs in to the sforce server and starts a client session.

Syntax

```
LoginResult = sfdc.login(string username, string password);
```


Usage

Use the login call to log in to the sforce logon server and start a client session. A client application *must* log in and obtain a session ID and server URL before making any other sforce API calls.

When a client application invokes the login call, it passes in a user name and password. Upon invocation, the sforce Web service authenticates the login and returns the session ID for the session, the user ID associated with the logged in user name, and an URL that points to the sforce Web service to use in all subsequent sforce API calls.

After logging in, a client application needs to:

- set the session ID in the SOAP header so that the sforce Web service can validate subsequent requests for this session
- specify the server URL as the target server for subsequent service requests

Development tools differ in the way you specify session headers and server URLs. For more information, see the documentation for your particular development tool.

Client applications do not need to explicitly log out to end the session. Sessions expire automatically after a period of inactivity, which can be configured (in the Setup section in the salesforce.com user interface) for your organization to be 30, 60, 120, or 240 minute intervals. Client applications should log in again after a session has expired or is close to expiring.

Sample Code—Java

The following sample Java code shows logging in to the sforce logon server, getting the login result, setting the target server URL to the returned URL, and setting the returned session ID into the session header for Axis.

```
private void login() {

   // Create binding object for sforce
   SoapBindingStub sfdc = (SoapBindingStub) new SforceServiceLocator().getSoap();

   // login
   LoginResult loginResult = sfdc.login("userName", "password");

   // Reset the SOAP endpoint to the returned server URL
   sfdc = (SoapBindingStub) new
   SforceServiceLocator().getSoap(new java.net. URL(loginResult.getServerUrl()));

   // Create a new session header object
   // add the session ID returned from the login
   _SessionHeader sh = new _SessionHeader();
   sh.setSessionId(loginResult.getSessionId());

   // Set the session header for subsequent call authentication
   sfdc.setHeader(new SforceServiceLocator().getServiceName().getNamespaceURI(),
"SessionHeader", sh);
}
```

Sample Code—C#

```
private void login()
{
   // Create service object for sforce
   SforceService sfdc = new SforceService();

   // Invoke the login call and save results in LoginResult
```



```
    LoginResult lr = sfdc.login("username","password");

    // Reset the SOAP endpoint to the returned server URL
    sfdc.Url = lr.serverUrl;

    // Create a new session header object
    // Add the session ID returned from the login
    sfdc.SessionHeaderValue = new SessionHeader();
    sfdc.SessionHeaderValue.sessionId = lr.sessionId;
}
```

Arguments

Name	Type	Description
username	string	Login user name.
password	string	Login password associated with the specified username.

Response

LoginResult

Fault

LoginFault

UnexpectedErrorFault

See Also

Sample SOAP Messages—login
Concepts on page 22

LoginResult

The login call returns a `LoginResult` object, which has the following properties:

Name	Type	Description
passwordExpired	boolean	Indicates whether the password used during the login attempt is expired (True) or not (False). If the password has expired, then the sforce Web service returns a valid sessionId, but the only allowable operation is the setPassword call.
serverUrl	string	URL of the sforce Web service that will process subsequent sforce API calls. Your client application needs to define the target server.

Name	Type	Description
sessionID	string	Unique ID associated with this session. Your client application needs to set this value in the session header.
userID	ID	ID of the user associated with the specified user name / password. If you want to retrieve information from your personal profile in the User object, you can pass this userID in the retrieve call. Alternatively, you can call getUserInfo to retrieve your personal profile information without this userID.

query

Executes a query against the specified object and returns data that matches the specified criteria.

Syntax

```
QueryResult = sfdc.query(string queryString);
```

Usage

Use the query call to retrieve data from an sforce API object. When a client application invokes the query call, it passes in a query expression that specifies the object to query, the fields to retrieve, and any conditions that determine whether a given object qualifies. For an extensive discussion about the syntax and rules used for queries, see sforce Object Query Language (SOQL) on page 28.

Upon invocation, the sforce Web service executes the query against the specified object, caches the results of the query on the sforce Web service, and returns a query response object to the client application. The client application can then use methods on the query response object to iterate through rows in the query response and retrieve information.

Your client application must be logged in with sufficient access rights to query individual objects within the specified object and to query the fields in the specified field list. For more information, see Factors that Affect Data Access on page 23.

Certain objects cannot be queried via the sforce API. To query an object via the query call, its object must be configured as queryable (`queryable=True`). To determine whether an object can be queried, your client application can invoke the describeSObject call on the object and inspect its `queryable` property.

The query response object contains up to 2,000 rows of data. If the query results exceed 2,000 rows, then the client application uses the queryMore call and a server-side cursor to retrieve additional rows in 2,000-row chunks. You can customize this option in the QueryOptions header, as described in Changing the Batch Size in Queries on page 32.

Queries that take longer than two minutes to process will be timed out. For timed out queries, the sforce Web service returns an ExceptionCode of QUERY_TIMEOUT. If a timeout occurs, refactor your query to return or scan a smaller amount of data.

When querying for fields of type Base64 (see Base64 Field Type on page 104), the query response object returns only one record at a time. You cannot alter this by changing the batch size of the query call.

Note

For multi-currency organizations, special handling is required when querying currency fields containing values in different currencies. For example, if a client application is querying PricebookEntry objects based on values in the `UnitPrice` field, and if the `UnitPrice` amounts are expressed in different currencies, then the query logic must handle this case correctly. For example, if the query is trying to retrieve the product codes of all products with a unit price greater than or equal to $10USD. the query expression might look something like this:


```
select Product2Id,ProductCode,UnitPrice from PricebookEntry where (UnitPrice >=
10 and CurrencyIsoCode="USD") or (UnitPrice >= 5.47 and CurrencyIsoCode="GBP") or
(UnitPrice >= 8.19 and CurrencyIsoCode="EUR")
```

Sample Code—Java

```
public void querySample() {

   QueryResult queryResult = null;
   // Set up query options. Set the max batch size to 3
   // so that we can exercise the queryMore call as well
   _QueryOptions queryOptions = new _QueryOptions();
   queryOptions.setBatchSize(new Integer(3));

   // Add the query options to the SOAP header
   binding.setHeader(new
SforceServiceLocator().getServiceName().getNamespaceURI(), "QueryOptions",
queryOptions);

   // Invoke the query call and save the results
   queryResult = binding.query("select FirstName, LastName from Contact");
   // Determine whether the query returned all the possible records
      if (queryResult.isDone()) {
      // Iterate through the records and process them
         for (int i = 0; i < queryResult.getRecords().length; i++) {
            Contact con = (Contact) queryResult.getRecords(i);
            String firstName = con.getFirstName();
            String lastName = con.getLastName();
            System.out.println("Contact " + (i + 1) + ": " + firstName + " " +
lastName);
         }
      }
      else {
         // Need to use queryMore call after processing
         // the first set of records from the query result
         while (queryResult.getRecords() != null) {
            for (int i = 0; i < queryResult.getRecords().length; i++) {
               Contact con = (Contact) queryResult.getRecords(i);
               String firstName = con.getFirstName();
               String lastName = con.getLastName();
               System.out.println("Contact " + (i + 1) + ": " + firstName + " " +
lastName);
            }
         // Invoke the queryMore call to get the next set of returned rows
         queryResult = binding.queryMore(queryResult.getQueryLocator());
      }
   }
}
```

Sample Code—C#

```
private void contactQuery()
{
   // Set the query options (Optional; default batch size is 2000)
   binding.QueryOptionsValue = new QueryOptions();
   binding.QueryOptionsValue.batchSize = 10;
   binding.QueryOptionsValue.batchSizeSpecified = true;
```



```
   // Invoke the query call and save the result in a QueryResult
   QueryResult qr = binding.query("select FirstName, LastName from contact where
MailingPostalCode = '94062'");

   // Get the returned records
   sObject[] records = qr.records;

   // Determine whether some records where returned
   if (records.Length > 0)
   {
      bool done = false;  // Use this for loop control
      while (done = false)
      {
         for (int i=0; i<records.Length; i++)
         {
            Contact contact = (Contact)records[i];
            System.Diagnostics.Trace.WriteLine(contact.FirstName + " " +
contact.LastName);
         }
         // Update the loop control
         done = qr.done;
         // Determine whether we need to retrieve another batch of result records
         if (done == false)
         {
             qr = binding.queryMore(qr.queryLocator);
             records = qr.records;
         }
    else
        {
            done = qr.done;
       }
   else
   {
      System.Diagnostics.Trace.WriteLine("no records matched criteria");
   }
}
```

Arguments

Name	Type	Description
queryString	string	Query string that specifies the object to query, the fields to return, and any conditions for including a specific object in the query. For more information, see sforce Object Query Language (SOQL) on page 28.

Response

QueryResult

Fault

MalformedQueryFault

InvalidSObjectFault

InvalidFieldFault

`UnexpectedErrorFault`

See Also

queryMore on page 82
Sample SOAP Messages—query
sforce Object Query Language (SOQL) on page 28
Concepts on page 22
Changing the Batch Size in Queries on page 32

QueryResult

The query call returns a `QueryResult` object, which has the following properties:

Name	Type	Description
queryLocator	QueryLocator	String. Used in queryMore for retrieving subsequent sets of objects from the query results, if applicable. Represents a server-side cursor. Note that an salesforce.com account can have up to five (5) query cursors open at a time.
done	boolean	Indicates whether additional rows need to be retrieved from the query results (False) using queryMore, or not (True). Your client application can use this value as a loop condition while iterating through the query results.
records	sObject[]	Array of sObjects representing individual objects of the specified object and containing data defined in the field list specified in the `queryString`.
size	int	Total number of rows retrieved in the query. Your client application can use this value to determine whether the query retrieved any rows (size > 0) or not (size = 0).

QueryLocator

In the QueryResult object returned by the query call, the queryLocator field contains a `QueryLocator` object that you will use in a subsequent queryMore call. Note that:

- You use a given `QueryLocator` only *once*. Each time you pass it in a queryMore call, the server returns a new `QueryLocator` in the QueryResult.
- `QueryLocator` objects expire automatically after 15 minutes of inactivity.

A `QueryLocator` represents a server-side cursor. A salesforce.com account can have up to five (5) query cursors open at a time. If five `QueryLocator` cursors are opened when a client application attempts open a new one, then the oldest of the five cursors is released.

queryMore

Retrieves the next batch of objects from a query.

Syntax

```
QueryResult = sfdc.queryMore(QueryLocator QueryLocator);
```


Usage

You use queryMore to process query calls that retrieve a large number of records (more than 2000) in the result set. The query call retrieves the first 2000 records and creates a server-side cursor that is represented in the queryLocator object. The queryMore call processes subsequent records in up to 2000-record chunks, resets the server-side cursor, and returns a newly generated QueryLocator. To iterate through records in the result set, you generally call queryMore repeatedly until all records in the result set have been processed (the `Done` flag is True).

Sample Code—Java

See the Sample Code—Java for the query call.

Sample Code—C#

See the Sample Code—C# for the query call.

Arguments

Name	Type	Description
queryLocator	QueryLocator	Represents the server-side cursor that tracks the current processing location in the query result set.

Response

QueryResult

Fault

`InvalidQueryLocatorFault`

`UnexpectedErrorFault`

See Also

query on page 79
Sample SOAP Messages—queryMore
Concepts on page 22
Changing the Batch Size in Queries on page 32

QueryResult

The queryMore call returns a `QueryResult` object, which has the following properties:

Name	Type	Description
queryLocator	QueryLocator	String. Used in subsequent queryMore calls for retrieving sets of objects from the query results, if applicable.
done	boolean	Indicates whether additional rows need to be retrieved from the query results (False) using another queryMore call, or not (True). Your client application can use this value as a loop condition while iterating through the query results.
records	sObject[]	Array of sObjects representing individual objects of the specified object and containing data defined in the field list specified in the `queryString`.
size	int	Total number of rows retrieved in the query. Your client application can use this value to determine whether the query retrieved any rows (size > 0) or not (size = 0).

QueryLocator

In the QueryResult object returned by the queryMore call, the queryLocator field contains a `QueryLocator` object that you will use in subsequent queryMore calls. Note that:

- You use a given `QueryLocator` only *once*. Each time you pass it in a queryMore call, the server returns a new `QueryLocator` in the QueryResult.
- `QueryLocator` objects expire automatically after 15 minutes of inactivity.

A `QueryLocator` represents a server-side cursor. A salesforce.com account can have up to five (5) query cursors open at a time. If five `QueryLocator` cursors are opened when a client application attempts open a new one, then the oldest of the five cursors is released.

retrieve

Retrieves one or more objects based on the specified object IDs.

Syntax

```
sObject[] result = sfdc.retrieve(string fieldList, string sObjectType, ID ids[]);
```

Usage

Use the retrieve call to retrieve individual objects from an sforce API object. The client application passes the list of fields to retrieve, the object, and an array of object IDs to retrieve. The retrieve call does *not* return objects that have been deleted.

In general, you use retrieve when you know in advance the IDs of the objects to retrieve. Use query instead to obtain objects when you do not know the IDs or when you want to specify other selection criteria.

Client applications can use retrieve to perform a client-side join. For example, a client application can run a query to obtain a set of Opportunity objects, iterate through the returned Opportunity objects, obtain the **accountId** for each Opportunity, and then call retrieve to obtain Account information for those **accountIds**.

Certain objects cannot be retrieved via the sforce API. To retrieve an object via the retrieve call, its object must be configured as retrieveable (`retrieveable=True`). To determine whether an object can be retrieved, your client application can invoke the describeSObject call on the object and inspect its `retrieveable` property.

Your client application must be logged in with sufficient access rights to retrieve individual objects within the specified object and to retrieve the fields in the specified field list. For more information, see Factors that Affect Data Access on page 23.

Sample Code—Java

```
private void retrieveSample() {
// Invoke the retrieve call and save results in an array of SObjects
SObject[] sObjects = binding.retrieve("Id, AccountNumber, Name, Website",
"Account", accounts);
// Verify that some objects were returned.
// Even though we began with valid object Ids,
// someone else might have deleted them in the meantime.
if (sObjects != null) {
   // Loop through the array and print out some properties
   for (int i=0;i<sObjects.length;i++) {
      // Cast the SObject into an Account object
      Account retrievedAccount = (Account)sObjects[i];
      System.out.println("Account: " + retrievedAccount.getId().getValue());
      System.out.println("    AccountNumber = " +
retrievedAccount.getAccountNumber());
      System.out.println("    Name          = " + retrievedAccount.getName());
      System.out.println("    Website       = " + retrievedAccount.getWebsite());
      }
   }
}
```

Sample Code—C#

```
private void retrieve()
{
   // Invoke retrieve call and save results in an array of SObjects
   sObject[] records = binding.retrieve("FirstName, LastName", "Contact", new
String[] {"", ""});

   // Iterate through the results
   for (int i=0;i<records.Length;i++)
   {
      Contact contact = (Contact)records[i];
      // Get the contact properties
      System.Diagnostics.Trace.WriteLine("Name is: " + contact.FirstName + " " +
contact.LastName);
   }
}
```


Arguments

Name	Type	Description
fieldList	string	List of one or more fields in the specified object, separated by commas. You must specify valid field names and must have read-level permissions to each specified field. The *fieldList* defines the ordering of fields in the result.
from	string	Object from which to retrieve data. The specified value must be a valid object for your organization. For a complete list of sforce objects, see List of sforce Objects on page 110.
ids	ID[]	Array of one or more IDs of the objects to retrieve. You can pass a maximum of 2000 object IDs to the retrieve call.

Response

Name	Type	Description
result	sObject[]	Array of one or more sObjects representing individual objects of the specified object. The number of sObjects returned in the array matches the number of object IDs passed into the retrieve call. If you do not have access to an object or if a passed ID is invalid, the array returns null for that object.

Fault

InvalidSObjectFault
InvalidFieldFault
UnexpectedErrorFault

See Also

Sample SOAP Messages—retrieve
Concepts on page 22

search

Executes a text search in your organization's data.

Syntax

```
SearchResult = sdfc.search(String searchString);
```


Usage

Use search to search for objects based on a search string. The search call supports searching custom object. For an extensive discussion about the syntax and rules used for text searches, see sforce Object Search Language (SOSL) on page 33.

Certain objects cannot be searched via the sforce API, such as Attachment objects. To search an object via the search call, its object must be configured as searchable (`searchable=True`). To determine whether an object can be searched, your client application can invoke the describeSObject call on the object and inspect its `searchable` property.

Sample Code—Java

```
private void searchSample() {

    SearchResult sr = null;
    //This search will look for a particular phone number in Contacts,
    //Leads and Accounts returning similar information for Contact
    //and Leads and just the name and phone number for the Accounts
      sr = binding.search("find {4159017000} " +
            "in phone fields " +
            "returning " +
            "contact(id, phone, firstname, lastname), " +
"lead(id, phone, firstname, lastname), " +
            "account(id, phone, name)");

     //Put the results into an array of SearchRecords
    SearchRecord[] records = sr.getSearchRecords();

    //Check the length of the returned array of records to see
    //if the search found anything
    if (records.length > 0) {
        //We are going to use vectors to hold the results
        Vector contacts = new Vector();
        Vector leads = new Vector();
        Vector accounts = new Vector();
        //We will go throught the results and determine what type
        //of object we found by using instanceof and add each record
        //to the correct vectory
        for (int i=0;i<records.length;i++){
            SObject record = (SObject)records[i].getRecord();
            if (record instanceof Contact) {
                contacts.add(record);
            } else if (record instanceof Lead){
                leads.add(record);
            } else if (record instanceof Account) {
                accounts.add(record);
            }
        }
        //we now have our results sorted into buckets of specific types
        //so we can report our findings
        if (contacts.size() > 0) {
           System.out.println("Found " + new Integer(contacts.size()).toString()
+ " contacts:");
            for (int i=0;i<contacts.size();i++){
                Contact c = (Contact) contacts.get(i);
               System.out.println(c.getFirstName() + " " + c.getLastName() + " -
" + c.getPhone());
            }
        }
```



```
        if (leads.size() > 0) {
           System.out.println("Found " + new Integer(leads.size()).toString() + "
leads:");
            for (int i=0;i<leads.size();i++){
                Lead l = (Lead) leads.get(i);
               System.out.println(l.getFirstName() + " " + l.getLastName() + " -
" + l.getPhone());
            }
        }
        if (accounts.size() > 0) {
            System.out.println("Found " + new Integer(accounts.size()).toString()
+ " accounts:");
            for (int i=0;i<accounts.size();i++){
                Account a = (Account) accounts.get(i);
                System.out.println(a.getName() + " - " + a.getPhone());
            }
        }
    } else {
        System.out.println("No records were found for the search.");
    }
}
```

Sample Code—C#

```
private void searchSample()
{
    sr = binding.search("find {4159017000} " +
        "in phone fields returning " +
        "contact(id, phone, firstname, lastname), " +
        "lead(id, phone, firstname, lastname), " +
        "account(id, phone, name)");

    sforce.SearchRecord[] records = sr.searchRecords;

    System.Collections.ArrayList contacts = new System.Collections.ArrayList();
    System.Collections.ArrayList leads = new System.Collections.ArrayList();
    System.Collections.ArrayList accounts = new System.Collections.ArrayList();

    if (records.Length > 0)
    {
        for (int i=0;i<records.Length;i++)
        {
            sforce.sObject record = records[i].record;
            if (record.GetType() == typeof(sforce.Contact))
            {
                contacts.Add(record);
            }
            else if (record.GetType() == typeof(sforce.Lead))
            {
                leads.Add(record);
            }
            else if (record.GetType() == typeof(sforce.Account) )
            {
                accounts.Add(record);
            }
            System.Diagnostics.Trace.WriteLine("out");
        }
        if (contacts.Count > 0)
        {
```



```
            Console.WriteLine("Found " + contacts.Count + " contacts:");
            for (int i=0;i<contacts.Count;i++)
            {
                sforce.Contact c = (sforce.Contact) contacts[i];
                Console.WriteLine(c.FirstName + " " + c.LastName + " - " +
c.Phone);
            }
        }
        if (leads.Count > 0)
        {
            Console.WriteLine("Found " + leads.Count + " leads:");
            for (int i=0;i<leads.Count;i++)
            {
                sforce.Lead l = (sforce.Lead) leads[i];
                Console.WriteLine(l.FirstName + " " + l.LastName + " - " +
l.Phone);
            }
        }
        if (accounts.Count > 0)
        {
            Console.WriteLine("Found " + accounts.Count + " accounts:");
            for (int i=0;i<accounts.Count;i++)
            {
                sforce.Account a = (sforce.Account) accounts[i];
                Console.WriteLine(a.Name + " - " + a.Phone);
            }
        }
    }
    else
    {
        Console.WriteLine("No records were found for the search.");
    }
}
```

Arguments

Name	Type	Description
searchString	String	Search string that specifies the text expression to search for, the scope of fields to search, the list of objects and fields to retrieve, and the maximum number of objects to return. For more information, see sforce Object Search Language (SOSL) on page 33.

Response

SearchResult

Fault

MalformedSearchFault

See Also

Concepts on page 22

SearchResult

The search call returns a `SearchResult` object, which has the following properties:

Name	Type	Description
`searchRecords`	SearchRecord[]	Array of SearchRecord objects, each of which contains an sObject.

update

Updates one or more existing objects in your organization's data.

Syntax

```
SaveResult[] = sfdc.update(sObject[] sObjects);
```

Usage

Use update to update one or more existing objects, such as individual accounts or contacts, in your organization's data. The update call is analogous to the UPDATE statement in SQL.

Rules and Guidelines

When updating objects, consider the following rules and guidelines:

Permissions. Your client application must be logged in with sufficient access rights to update individual objects (as well as individual fields inside that object) within the specified object. For more information, see Factors that Affect Data Access on page 23.

Special Handling. Certain objects—and certain fields within those objects—require special handling or permissions. For example, you might also need permissions to access this object's parent object. Before you attempt to update a particular object, be sure to read its description in Chapter 5: sforce Objects on page 101.

Updateable Objects. Certain objects cannot be updated via the sforce API. To update an object via the update call, its object must be configured as updateable (`updateable=True`). To determine whether an object can be updated, your client application can invoke the describeSObject call on the object and inspect its `updateable` property.

Required Fields. When updating required fields, you must supply a value—you cannot set the value to `null`. For more information, see Required Fields on page 107.

ID Fields. You cannot update fields with "Id" in the name, such as reference fields that point to other objects (for example, **`CaseId`** or **`OpportunityId`**). Such fields are analogous to a primary or foreign key field in SQL databases.

Automatically Updated Fields. The sforce Web service updates certain fields automatically, such as **`LastModifiedDate`**, **`LastModifiedById`**, and **`SystemModstamp`**. You cannot explicitly specify these values in your update call.

Resetting Values to null. To reset a field value to `null`, you add the field name to the `fieldsToNull` array in the sObject. You cannot set required fields (`nillable=false`) to `null`.

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 specified for your programming language and development tool (your development tool will handle the appropriate mapping of data types in SOAP messages).

String Values. When storing values in string fields, the sforce Web service trims any leading and trailing whitespace. For example, if the value of a name field is entered as `"  ABC Company "`, then the value is stored in the database as `"ABC Company"`.

Assignment Rules. When updating Case or Lead objects, your client application can set AssignmentRuleHeader options to have the case or lead automatically assigned to one or more users based on assignment rules configured in the salesforce.com user interface. For more information, see Case on page 126 or Lead on page 141.

Maximum Number of Objects Created. Your client application can change up to 200 individual objects in a single update call. If an update request exceeds 200 objects, then the entire operation fails.

Basic Steps for Updating Objects

Updating objects involves the following basic steps:

1. Determine the ID of each object that you want to update. For example, you might call query to retrieve a set of objects (with their IDs), based on specific criteria, that you would want to update. If you know the ID of the object that you want to update, you can call retrieve instead.
2. For each object, populate its fields with the data that you want to update.
3. Construct an `sObject[]` array and populate that array with the objects that you want to update. All objects *must* be of the same object.
4. Call update, passing in the `sObject[]` array.
5. Process the results in the `SaveResult[]` object to verify whether the objects have been successfully updated.

Sample Code—Java

```
public void updateAccountSample() {

   // Create an array of SObjects to send to the update method
   SObject[] updates = new SObject[2];

   // This account could also be from the results of a retrieve or query call
   Account updateAccount = new Account();
   updateAccount.setId(new ID("001x00000000JerAAE"));
   updateAccount.setName("New Account Name from Update Sample");
   updates[0] = updateAccount;

   Account updateAccount2 = new Account();
   updateAccount2 = new Account();
   updateAccount2.setId(new ID("001x00000000JesAAE"));
   updateAccount2.setWebsite("www.website.com");
   updates[1] = updateAccount2;

   // Invoke the update call and save the results
   SaveResult[] saveResults = binding.update(updates);
   print("\nPress the RETURN key to continue...", false);
}
```

Sample Code—C#

```
private void update()
{
   // You would typically retrieve an SObject, modify its properties,
   // and then send the objects up in an array. For this sample,
```



```
    // we create a new contact object to update by setting
    // the id to a valid contact id
    Contact contact = new Contact();
    contact.Id = "";  // This should be a valid ID
    contact.MailingCity = "new city";
    contact.MailingPostalCode = "98776";

   // Invoke the update call, saving the results in SaveResult
    SaveResult[] sr = binding.update(new sObject[]{contact});

    // The SaveResult should never be empty
    for (int i=0;i<sr.Length;i++)
    {
       // Determine whether the row update succeeded
       if (sr[i].success)
       {
          // Get the ID of the updated row
    System.Diagnostics.Trace.WriteLine(sr[i].id);
       }
       else
       {
          // Iterate through the errors
          Error[] errors = sr[i].errors;
          for (int j=0;j<errors.Length;j++)
          {
             System.Diagnostics.Trace.WriteLine(errors[j].message);
          }
       }
    }
}
```

Arguments

Name	Type	Description
sObjects	sObject[]	Array of one or more objects (maximum of 200) to update. sforce Web service updates these objects in array index order.

Response

SaveResult[]

Fault

InvalidSObjectFault

UnexpectedErrorFault

See Also

Sample SOAP Messages—update
Concepts *on page 22*

SaveResult

The update call returns an array of `SaveResult` objects. Each element in the `SaveResult` array corresponds to the sObject[] array passed as the `sObjects` parameter in the update call. For example, the object returned in the first index in the `SaveResult` array matches the object specified in the first index of the sObject[] array.

A `SaveResult` object has the following properties:

Name	Type	Description
id	ID	ID of an sObject that you attempted to update.
success	boolean	Indicates whether the update call succeeded (True) or not (False) for this object.
errors	Error[]	If an error occurred during the update call, an array of one or more Error objects providing the error code and description.

CHAPTER 4: sforce Utility API Calls

This topic describes sforce Web services API calls that your client applications can invoke to obtain the server timestamp, user information, and change user passwords. For a complete list of all sforce API calls, see List of sforce API Calls on page 43.

The following table lists the sforce utility API calls described in this topic:

Table 4: Supported Utility Calls in the sforce API

Task / Call	Description
getServerTimestamp	Retrieves the current system timestamp from the sforce Web service.
getUserInfo	Retrieves personal information for the user associated with the current session.
resetPassword	Changes a user's password to a server-generated value.
setPassword	Sets the specified user's password to the specified value.

getServerTimestamp

Retrieves the current system timestamp (GMT) from the sforce Web service.

Syntax

```
dateTime timestamp = sfdc.getServerTimestamp();
```

Usage

Use getServerTimestamp to obtain the current system timestamp from the sforce Web service. You might do this if, for example, you need to use the exact timestamp for timing or data synchronization purposes. When you create or update an object, the sforce Web service uses the system timestamp to update the **`CreatedDate`** and **`LastModifiedDate`** fields, respectively, in the object.

The getServerTimestamp call always returns the timestamp in GMT. However, your local system might automatically display the results in your local time based on your time zone settings.

Note Development tools differ in the way that they handle time data. Some development tools report the local time, while others report only the GMT time. To determine how your development tool handles time values, refer to its documentation.

Sample Code—Java

```
public void getServerTimestampSample() {
   // Invoke the getServerTimestamp call and save the results
   GetServerTimestampResult serverTimestampResult = sfdc.getServerTimestamp();
   System.out.println("Server Timestamp: " +
serverTimestampResult.getTimestamp());
```



```
}
```

Sample Code—C#

```
private void getServerTimeStamp()
{
   // Invoke the getServerTimeStamp call and save the results
   GetServerTimestampResult ts = binding.getServerTimestamp();
   // Write the server timestamp to the diagnostics window
   System.Diagnostics.Trace.WriteLine(ts.timestamp.ToUniversalTime);
}
```

Arguments

None.

Response

Name	Type	Description
timestamp	dateTime	System timestamp of the sforce Web service when the getServerTimestamp call was executed.

Fault

UnexpectedErrorFault

See Also

Sample SOAP Messages—getServerTimestamp
sforce Utility API Calls on page 94

getUserInfo

Retrieves personal information for the user associated with the current session.

Syntax

```
GetUserInfoResult result = sfdc.getUserInfo();
```

Usage

Use getUserInfo to obtain personal information about the currently logged in user. The getUserInfo call is a convenience API call that retrieves and aggregates common profile information that your client application can use for display purposes, performing currency calculations, and so on.

The getUserInfo call applies only to the user name under which your client application has logged in. To retrieve additional personal information not found in the GetUserInfoResult object, you can call retrieve on the User object and pass in the `userID` returned by this call. To retrieve personal information about other users, you could call retrieve (if you know their user ID) or query on the User object.

Sample Code—Java

```
public void getUserInfoSample() {

   GetUserInfoResult getUserInfoResult = null;

   // Invoke the getUserInfo call
   getUserInfoResult = binding.getUserInfo("fil@fil.com");

   // Display the returned user information
   System.out.println("User's currency symbol: " +
                      getUserInfoResult.getCurrencySymbol());
   System.out.println("User's organization name: " +
                      getUserInfoResult.getOrganizationName());
   System.out.println("User's default currency code: " +
                      getUserInfoResult.getUserDefaultCurrencyIsoCode());
   System.out.println("User's email: " + getUserInfoResult.getUserEmail());
   System.out.println("User's full name: " +
getUserInfoResult.getUserFullName());
   System.out.println("User's user id: " + getUserInfoResult.getUserId());
   System.out.println("User's language: " + getUserInfoResult.getUserLanguage());
   System.out.println("User's locale: " + getUserInfoResult.getUserLocale());
   System.out.println("User's timezone: " + getUserInfoResult.getUserTimeZone());
   System.out.println("User's org is multi currency: " +
                      getUserInfoResult.isOrganizationMultiCurrency());
}
```

Sample Code—C#

```
private void getUserInfo()
{
   // Invoke getUserInfo call and save the results in getUserInfoResult
   GetUserInfoResult ui = binding.getUserInfo();
   // Get some of the user information
   String orgName = ui.organizationName;
   String userFullName = ui.userFullName;
}
```

Arguments

None.

Response

GetUserInfoResult

Fault

UnexpectedErrorFault

See Also

Sample SOAP Messages—getUserInfo
sforce Utility API Calls on page 94

GetUserInfoResult

The getUserInfo call returns a `GetUserInfoResult` object, which has the following properties:

Name	Type	Description
currencySymbol	string	Currency symbol to use for displaying currency values. Applicable only when **organizationMultiCurrency** is False.
organizationId	ID	Organization ID. Allows third-party tools to uniquely identify individual organizations in salesforce.com, which is useful for retrieving billing or organization-wide setup information.
organizationMultiCurrency	boolean	Indicates whether the user's organization uses multiple currencies (True) or not (False).
organizationName	string	Name of the user's organization or company.
userDefaultCurrencyIsoCode	string	Default currency ISO code. Applicable only when **organizationMultiCurrency** is True. When the logged in user creates any objects that have a currency ISO code, the server uses this currency ISO code if it is not explicitly specified in the create call.
userEmail	string	User's email address.
userFullName	string	User's full name.
userID	ID	User ID.
userLanguage	string	User's language.
userLocale	string	User's locale (language and country).
userTimeZone	string	User's time zone.

resetPassword

Changes a user's password to a server-generated value.

Syntax

```
string password = sfdc.resetPassword(ID userID);
```

Usage

Use resetPassword to request that the sforce Web service change a user's password and return the server-generated password string. Use setPassword instead if you want to set the password to a specific value.

Your client application must be logged in with sufficient access rights to change the password for the specified user. For more information, see Factors that Affect Data Access on page 23.

Sample Code—Java

```
public void resetPasswordSample() {
```



```
   // Specify the user ID of the password to reset
   String idToReset = "005x00000001ZPH";

   // Invoke the resetPasswordResult call
   ResetPasswordResult resetPasswordResult = binding.resetPassword(new
ID(idToReset));

   // Display the new server-generated password
   System.out.println(resetPasswordResult.getPassword());
}
```

Sample Code—C#

```
private void resetPassword()
{
   // Invoke resetPassword call and save results in ResetPasswordResult
   ResetPasswordResult rpr = binding.resetPassword("userID");
   // Get the generated password
   System.Diagnostics.Trace.WriteLine(rpr.password);
}
```

Arguments

Name	Type	Description
userID	ID	ID of the user whose password you want to reset.

Response

Name	Type	Description
password	string	New password generated by the sforce Web service.

Fault

InvalidIdFault

UnexpectedErrorFault

See Also

setPassword
Sample SOAP Messages—resetPassword
sforce Utility API Calls on page 94

setPassword

Sets the specified user's password to the specified value.

Syntax

```
SetPasswordResult setPasswordResult = sfdc.setPassword(ID userID string
password);
```

Usage

Use setPassword to change a user's password to a value that you specify. For example, a client application might prompt a user to specify a different password, and then invoke setPassword to change the user's login password. Use resetPassword instead if you want to reset the password with an sforce Web service-generated value.

Your client application must be logged in with sufficient access rights to change the password for the specified user. For more information, see Factors that Affect Data Access on page 23.

Sample Code—Java

```
public void setPasswordSample() {

   // Specify the userID and new password
   String idToReset = "005x00020001ZPH";
   String newPassword = "bigsecret";

   // Invoke the setPassword call
   SetPasswordResult setPasswordResult = binding.setPassword(new ID(idToReset),
newPassword);
   // If the call fails, an exception is raised; otherwise, the return is empty.
}
```

Sample Code—C#

```
private void setPassword()
{
   // Invoke setPassword call; returns nothing if successful
   binding.setPassword("userid", "newpassword");
}
```

Arguments

Name	Type	Description
userID	ID	ID of the user whose password you want to reset.
password	string	New password to use for the specified user.

Response

None.

Fault

InvalidIdFault

UnexpectedErrorFault

See Also

resetPassword on page 97
Sample SOAP Messages—setPassword
sforce Utility API Calls on page 94

CHAPTER 5: sforce Objects

This topic describes all of the possible objects defined in the sforce Web services API. It contains the following sections:

CONCEPTS

This section describes the following concepts:

About sforce API Objects

In the sforce API, *objects* are data entities that represent your organization's information. For example, the Account object represents accounts—companies and organizations involved with your business, such as customers, partners, and competitors. To describe a particular occurrence of an object (such as a specific account that is represented by an Account object), this document uses the term *object instance*. An object instance is analogous to a row in a database table.

Access to Objects

While this topic describes all of the objects available in the sforce API, your applications are able to work with only the objects that you are authorized to access. Programmatic access to objects is determined by the objects that are defined in your Enterprise WSDL file, your organization configuration, your security access (which is configured by your organization's system administrator in your personal profile), and your data sharing model. For more information, see Factors that Affect Data Access on page 23.

Field Types

The sforce API uses the following field types:

Table 5: Field Types Used in the sforce API

Field Type	What the Field Contains
`string`	String values. See String Field Type on page 103.
`boolean`	Boolean (True / False) values. See Boolean Field Type on page 103.
`int`	Integer (int) values. See Int Field Type on page 103.
`double`	Double values. See Double Field Type on page 103.
`date`	Date values. See Date Field Type on page 103.
`datetime`	Date and time values. See DateTime Field Type on page 103.
`base64`	Base64-encoded arbitrary binary data (of type base64Binary). Used for Attachment, Document, and Scontrol objects. See Base64 Field Type on page 104.
`id`	Primary key field for the object. See Id Field Type on page 104.
`reference`	Cross-references to a different sforce object. Analogous to a foreign key field in SQL. See Reference Field Type on page 104.
`currency`	Currency values. See Currency Field Type on page 104.
`textarea`	String that is displayed as a multi-line text field. See Textarea Field Type on page 105.
`percent`	Percentage values. See Percent Field Type on page 105.
`phone`	Phone numbers. Values can include alphabetic characters. Client applications are responsible for phone number formatting. See Phone Field Type on page 105.
`url`	URL values. Client applications should commonly display these as hyperlinks. See URL Field Type on page 105.
`email`	Email addresses. See Email Field Type on page 105.
`picklist`	Picklists, which include a set of enumerated values from which one value can be selected. See Picklist Field Type on page 105.
`multipicklist`	Multi-select picklists, which include a set of enumerated values from which multiple values can be selected. See Multi-Select Picklist Field Type on page 106.
`combobox`	Combobox, which include a set of enumerated values and allow the user to specify a value not in the list. See Combobox Field Type on page 106.

These field types extend the primitive data types, which are described in Primitive Data Types on page 248. While many of these field types follow common data typing conventions that are made explicit in their metadata, certain field types have unique characteristics that you need to understand before using them in your client application.

These field types apply to both predefined fields and custom fields. They are enumerated in the `type` field of the Field type, which is described in the `fields` property of the DescribeSObjectResult.

Note

Some numeric fields have precision and scale limits. In addition, certain text fields have length restrictions. These restrictions are enforced when you create or update objects. However, the sforce API may return data that does not meet these restrictions.

String Field Type

String fields (string) contain text and some have length restrictions depending on the data being stored. For example, in the Contact object, the **FirstName** field is 40 characters, the **LastName** field is 80 characters, the **MailingStreet** is 255 characters.

Boolean Field Type

Boolean (boolean) fields have either of two values:

- True (or 1)
- False (or 0)

Int Field Type

Integer fields (int) are numbers that contain no fractional portion (digits to the right of a decimal place), such as the NumberOfEmployees in an Account. For integer fields, the digits field specifies the maximum number of digits that an integer can have.

Double Field Type

Double fields (double) can contain fractional portions (digits to the right of the decimal place), such as **ConversionRate** in CurrencyType. In the sforce API, all non-integer values (such as Currency Field Type and Percent Field Type) are of type double. For double fields, the following restrictions might exist:

Table 6: Limitations on Double Fields

Fields	Description
scale	Maximum number of digits to the right of the decimal place.
precision	Total number of digits, including those to the left and the right of the decimal place

The maximum number of digits to the left of the decimal place is equal to precision minus scale. In the salesforce.com user interface, precision is defined differently—it is the maximum number of digits allowed to the left of the decimal place.

Date Field Type

Date fields (date) contain date values, such as **ActivityDate** in the Event object. Unlike dateTime fields, date fields contain no time value—the time portion of a date field is not relevant and is always set to midnight in the GMT/UTC time zone.

DateTime Field Type

Note

Development tools differ in the way that they handle time data. Some development tools report the local time, while others report only the GMT time. To determine how your development tool handles time values, refer to its documentation.

DateTime (dateTime) fields handle date/time values (timestamps), such as **ActivityDateTime** in the Event object or the **CreatedDate**, **LastModifiedDate**, or **SystemModstamp** in many sforce objects. Regular Date/Time fields are full timestamps with a precision of one second. They are always transferred in the GMT/UTC time zone. In your client application, you might need to translate the timestamp to or from a local time zone.

Note

The Event object has a **DurationInMinutes** field that specifies the number of minutes for an event. Even though this is a temporal value, it is an integer type—not a dateTime type.

Base64 Field Type

Base64 fields contain base64-encoded arbitrary binary data (of type base64Binary). These fields are used for storing binary files in Attachments, Documents, and Scontrol objects. In these objects, the **Body** or **Binary** field contains the (base64 encoded) data, while the **BodyLength** field defines the length of the data in the **Body** or **Binary** field. In the Document object, you can specify an URL to the document instead of storing the document directly in the record.

Id Field Type

All objects have an **Id** field (of type ID) that uniquely identifies each record in that type. This is analogous to the concept of a primary key in relational databases. The value in the **Id** field is assigned by the sforce Web service when the record is originally created (create call) to ensure that it is globally unique. This value remains unchanged over the entire lifetime of the record. Its value contains a three-character code that identifies the object type, which client applications can retrieve via the describeSObject call (see keyPrefix on page 66).

Some API calls, such as retrieve and delete, accept an array of **Id**s as parameters—each array element uniquely identifies the row to retrieve or delete. Similarly, the update call accepts an array of sObjects—each sObject contains an **Id** field that uniquely identifies the sObject.

Note

In .NET, the values in Id fields are treated simply as strings rather than as values of a special Id type.

Reference Field Type

A reference field contains an **Id** value (of type ID) that points to a unique record (usually the parent record) on another object. This is analogous to the concept of a foreign key in relational databases. The name of a reference field ends, by convention, with the letters **Id** (such as **CaseId** or **OpportunityId**). For example, in the OpportunityCompetitor object, the **OpportunityId** field is a reference field that points to the Opportunity object. It contains an **Id** value that uniquely identifies an Opportunity record.

In some cases, an object can refer to another object of its same type. For example, an Account can have a parent link that points to another Account.

The Event and Task objects both have **WhoId** and **WhatId** cross-reference ID fields. Each of these cross-reference fields can point to one of several other objects. The **WhoId** field can point to a Contact or Lead, and the **WhatId** field can point to an Account, Opportunity, Campaign, or Case. In addition, if the **WhoId** field refers to a Lead, then the **WhatId** field must be empty.

You can describe and query each cross-referenced object. When you query a cross-reference ID field, it returns an object ID of the appropriate type. You can then query that ID to get additional information about the object, using the ID in the **id** field for that query.

The cross-reference ID field value is either

- a valid record in your organization, or
- an empty value, which indicates an empty reference

The cross-reference ID field value, if non-null, is guaranteed to be an object in your organization. However, it is not guaranteed that you can query that object. Users with the "View All Data" permission can always query that object. Other users may be restricted from viewing or editing the referenced object.

When specifying a value for a cross-reference ID field in a create or update call, the value must be a valid value of type ID, and the user must have appropriate access to that object. The exact requirements vary from field to field.

Currency Field Type

Currency fields contain currency values, such as the **ExpectedRevenue** field in a Campaign, and are defined as type double.

For organizations that have the multi-currency option enabled, the **CurrencyISOCode** field is defined for any object that can have currency fields. The **CurrencyISOCode** field and currency fields are linked in a special way. On any specific record, the **CurrencyISOCode** field defines the

currency of that record, and thus, the values of all currency fields on that record will be expressed in that currency.

For most cases, clients do not need to consider the linking of the **CurrencyISOCode** field and the currency fields on an object. However, clients may need to consider the following:

- The **CurrencyISOCode** field exists only for those organizations that have enabled multi-currency support.
- When displaying the currency values in a user interface, it is preferred to prepend each currency value with its **CurrencyISOCode** value and a space separator.
- The **CurrencyISOCode** field is a restricted picklist field. The set of allowable values, defined in the CurrencyType object, can vary from organization to organization. Attempting to set it to a value that is not defined for an organization causes the operation to be rejected.
- If you update the **CurrencyISOCode** field on an object, it implicitly converts all currency values on that object to the new currency code, using the conversion rates that are defined for that organization in the salesforce.com user interface. If you specify currency values in that same update call, the new currency values you specify are interpreted in the new **CurrencyISOCode** field value, without conversion.

To perform currency conversions, client applications can look up the **CurrencyISOCode** in the CurrencyType object.

Textarea Field Type

Textarea fields contain text that can be longer than 4000 bytes. Unlike string fields, textarea fields cannot be specified in the WHERE clause of a `queryString` of a query call. To filter records on this field, you must do so while processing records in the QueryResult. For fields with this restriction, its **filterable** field in the Field type (described in the `fields` property of the DescribeSObjectResult) is False.

Percent Field Type

Percent fields contain percent values. Percent fields are defined as type double.

Phone Field Type

Phone fields contain phone numbers, which can include alphabetic characters. Client applications are responsible for phone number formatting.

URL Field Type

URL fields contain URLs. Client applications are responsible for specifying valid and properly formatted URLs in create and update calls.

Email Field Type

Email fields contain email addresses. Client applications are responsible for specifying valid and properly formatted email addresses in create and update calls.

Picklist Field Type

Picklist fields contain a list of one or more items from which a user chooses a single item. One of the items can be configured as the default item.

In the Field object associated with the DescribeSObjectResult, the `restrictedPicklist` field defines whether it is a restricted picklist or not. The sforce API does not enforce the list of values for advisory (unrestricted) picklist fields on create or update. When inserting a unrestricted picklist field that does not have a PicklistEntry, the system creates an "inactive" picklist value. This value can be promoted to an "active" picklist value by adding the picklist value in the salesforce.com user interface.

In the Field object associated with the DescribeSObjectResult, the `picklistValues` field contains an array of items (PickListEntry objects). Each PickListEntry defines the item's label, value, and whether it is the default item in the picklist (a picklist has no more than one default value).

Enumerated fields support localization of the labels to the language of the user. For example, for the `ForecastCategory` field on an Opportunity, the value "Omitted" may be translated to various languages. The enumerated field values are fixed and do not change with a user's language. However, each value may have a specified "label" field that provides the localized label for that value. You must always use the value when inserting or updating a field. The query call always returns the value, not the label. The corresponding label for a value in the DescribeSObjectResult should be used when displaying the value to the user in any user interface.

The sforce API supports the retrieval of the certain picklists in the following objects: CaseStatus, ContractStatus, LeadStatus, OpportunityStage, PartnerRole, SolutionStatus, TaskPriority, and TaskStatus. Each object represents a value in the respective picklist. These picklist entries always specify some other piece of information, such as whether the status is converted, etc. Your client application can invoke the query call on any of these objects (such as CaseStatus) to retrieve the set of values in the picklist, and then use that information while processing other objects (such as Cases) to determine more information about those objects (such as a given case). These objects are read-only via the sforce API. To modify items in picklists, you must use the salesforce.com user interface.

Multi-Select Picklist Field Type

Multi-select picklist fields contain a list of one or more items from which a user can choose multiple items. One of the items can be configured as the default item. Selections are maintained as a string containing a series of attributes delimited by semi-colons. For example, a query might return the values of a multi-value picklist as "first value; second value; third value".) For information on querying multi-select picklists, see Querying Multi-Select Picklists on page 32.

Combobox Field Type

A combobox is a picklist that also allows users to type a value that is not already specified in the list. A combobox is defined as a string value.

ID Fields

In the sforce API, all sforce objects, including custom objects, have a unique ID field (see Id Field Type on page 104) that uniquely identifies the object. This ID field is analogous to a primary key in relational databases. The ID field is automatically generated by salesforce.com when the object is created. Its value contains a three-character code that identifies the object type, which client applications can retrieve via the describeSObject call (see `keyPrefix` on page 66).

In addition, certain sforce objects, including custom objects, have one or more reference fields (see Reference Field Type on page 104) that contains the `Id` value (of type ID) of another, related object (such as a parent object). Reference fields are analogous to the concept of a foreign key in relational databases. The name of a reference field ends, by convention, with the letters `Id` (such as `CaseId` or `OpportunityId`).

System Fields

The following fields are read-only fields commonly found in sforce objects. The sforce Web service updates these fields automatically.

Table 7: Common Fields in sforce Objects

Field	Data Type	Description
`Id`	ID	Globally unique ID of this field. See Id Field Type on page 104.
`CreatedById`	ID	ID of the User who created this object. Read-only.
`CreatedDate`	dateTime	Date and time when this object was created. Read-only.
`LastModifiedById`	ID	ID of the User who last updated this object. Read-only.

Table 7: Common Fields in sforce Objects (Continued)

Field	Data Type	Description
`LastModifiedDate`	dateTime	Date and time when this object was last modified by a user. Read-only.
`SystemModstamp`	dateTime	Date and time when this record was last modified by a user or by a workflow process (such as a trigger). Read-only.

Required Fields

Required fields must have a non-`null` value. This rule affects the create and update calls:

- When a client application invokes the create call, salesforce.com automatically populates the data for certain required fields (such as system fields and the object ID fields). Similarly, if a required field has a default value (its **`defaultedOnCreate`** attribute is set to True, as described in `defaultedOnCreate` on page 68), then salesforce.com implicitly assigns a value for this field when the object is created, even if a value for this field is not explicitly passed in on the create call. For all other required fields, such as Id fields that are analogous to foreign keys in SQL (see Reference Field Type on page 104), a client application must explicitly assign a value when the object is created (it cannot be null).
- When a client application invokes the update call, a required field cannot be set to `null`. Many required fields cannot be changed in an update call.

For more information about the special handling of required fields for particular objects, see the documentation for such objects later in this topic.

Relationships Among sforce Objects

Certain sforce objects, including custom objects, have the following kinds of relationships with other objects:

- **Master-Detail (1:1)**—When a record in the master object is deleted, the related records in the detail object are also deleted. The security settings for the master record also control the detail record. The lookup field is the field linking the two objects. You can not create a master-detail relationship to users or leads.
- **Lookup (1:n)**—This type of relationship has no effect on deletion or security, and the lookup field is not required. When you define a lookup relationship for a custom object, data from the custom object can appear as a custom related list on page layouts for the other object.

These relationships are already defined for standard sforce objects. For custom objects, you configure these relationships in the salesforce.com user interface. For detailed information, see the salesforce.com user interface online help.

Custom Objects and Custom Fields

In the salesforce.com user interface, organizations can extend their salesforce.com data by defining custom objects and, for certain objects, custom fields.

Custom Objects

Custom objects are custom salesforce.com tables that allow you to store information unique to your organization. For custom objects, the **`custom`** flag—a boolean field in the DescribeSObjectResult—is `true`. Custom objects are defined in the salesforce.com user interface—client applications cannot create custom objects via the sforce API. However, client applications with sufficient permissions can invoke other API calls on custom objects.

Custom Fields

Organizations can also define custom fields for standard or custom objects. For custom fields, the **`custom`** flag—a boolean field in the Field object—is `true`. Custom fields are defined in the

salesforce.com user interface—client applications cannot define custom fields via the sforce API. For the most part, client applications do not need to know whether a field is a standard field or a custom field.

The following sforce API objects support custom fields:

- Asset
- Account
- Case
- Campaign
- Contact
- Contract
- Event
- Lead
- Opportunity
- OpportunityLineItem
- Product2
- Solution
- Task
- User

Note that all numeric custom fields are handled as type double. For more information, see Double Field Type on page 103.

Naming Conventions for Custom Objects and Custom Fields

Custom objects and custom fields have an associated name field that is defined by your salesforce.com administrator. Custom objects must have unique names within your organization, and custom fields must have unique names within the same object. In your WSDL file, custom object and field names have a `__c` suffix, such as `myCustomObject__c` and `myCustomField__c`.

Relationships Among Custom Objects

Note

Custom objects related to other objects behave just like standard sforce objects, as described in Relationships Among sforce Objects on page 107. For example, cascading deletes are supported in custom objects in a Master-Detail relationship.

The following objects can parent custom objects. An object with an asterisk next to it cannot participate in a Master-Detail relationship.

- Account
- Campaign
- Case
- Contact
- Contract
- Lead (*)
- Opportunity
- Product [Deprecated](*)
- Solution
- User (*)

The following table summarizes whether (X=true) a given sforce object:

- can be the master in master-detail relationship with a custom object (master-detail relationships involve cascading deletes and sharing rules controlled by the parent)

- can be a lookup (1:n) in a custom object (whether a lookup from the custom object can be performed on the sforce object)
- can be extended with custom fields

sforce Object	Master-Detail	Lookup	Custom Fields
Account	X	X	X
Campaign	X	X	X
Case	X	X	X
Contact	X	X	X
Contract	X	X	X
Event			X
Lead		X	X
Opportunity	X	X	X
Product2		X	X
Solution	X	X	X
Task			X
User		X	X

Common Fields in sforce Objects

Several fields are found in all sforce objects. There are few exceptions.

OwnerID Fields. sforce objects have an **`ownerID`** field that is an object reference field to the user that owns that object. Ownership is an important concept that affects the security model and has other implications throughout the system. Any user can query the owner field for any record they can access. However, the **`ownerID`** field has the following limitations when being set:

- For most users and most objects, the **`ownerID`** field cannot be set directly upon insert. It is implicitly set to the current user when inserting an object.
- When creating or updating a Case or Lead, a client application (that is logged in with permissions to transfer a record) can set the **`ownerID`** field to any valid User in the organization or to any valid queue of the appropriate type in the organization.
- Updating the **`ownerID`** field via the API changes only the owner of that record. The change of ownership does not cascade to associated records as it does when you transfer record ownership in the application.
- Updating the **`ownerID`** field on an account deletes the existing sharing information and reapplies the default sharing model and autoshare rules.

RecordTypeId fields. Record types are used to offer different business processes and subsets of picklist values to different Users based on their particular Profiles.

Record types are configured in the salesforce.com user interface. The RecordType object is read-only in the sforce API. A client application can retrieve the list of valid record type IDs for a given object by calling query on the RecordType object.

The **`RecordTypeId`** field can contain the ID of the RecordType associated with an sforce object. Client applications can set this field in create or update calls on these objects. If specified in a create or update call (the default is `nill`), the record type ID must refer to a valid record type for that object. The `RecordTypeId` field is associated with to the following objects: Account, Campaign, Case, Contact, Contract, Lead, Opportunity, and Solution.

Note

The `RecordTypeId` field will appear in your WSDL file only if at least one record type is configured for your organization in the salesforce.com user interface.

CurrencyIsoCode. For organizations that have multi-currency enabled, the **`CurrencyIsoCode`** field contains the string representation of the currency ISO code associated with currency values in the object. Note that the User object also has a **`DefaultCurrencyIsoCode`** field, which is the default currency for that user. For example, a user in France could have a **`DefaultCurrencyIsoCode`** set to Euros, and that would be their default currency in the application. However, the User object could have currency custom fields stored in a different currency.

LIST OF SFORCE OBJECTS

Table 8: Supported Objects in the sforce API

Object	Description
Account	Represents an individual account, which is an organization involved with your business (such as customers, competitors, and partners).
AccountContactRole	Represents the role that a given Contact plays on an Account.
AccountShare	Represents a sharing entry on an Account.
AccountTeamMember	Represents a User who is a member of an Account team.
Approval	Represents an approval request for a Contract.
Asset	Represents an asset (such as product previously sold and installed) owned by an Account or Contact.
AssignmentRule	Represents an assignment rule associated with a Case or Lead.
Attachment	Represents a file that a User has uploaded and attached to a parent object.
BusinessProcess	Represents a business process.
Campaign	Represents and tracks a marketing campaign, such as a direct mail promotion, webinar, or trade show.
CampaignMember	Represents the association between a Campaign and either a Lead or Contact.
Case	Represents a case, which is a customer issue such as a customer's feedback, problem, or question.
CaseComment	Represents a comment that provides additional information about the associated Case.
CaseHistory	Represents historical information about changes that have been made to the associated Case.
CaseSolution	Represents the association between a particular Case and a particular Solution.
CaseStatus	Represents the status of a Case, such as New, On hold, In Process, and so on.

Table 8: Supported Objects in the sforce API (Continued)

Object	Description
Contact	Represents a contact, which is an individual associated with your Accounts.
Contract	Represents a contract (a business agreement) associated with an Account.
ContractContactRole	Represents the role that a given Contact plays on a Contract.
ContractStatus	Represents the status of a Contract, such as Draft, In Approval, Activated, Terminated, or Expired.
CurrencyType	Represents the currencies used by an organization for which the multi-currency feature is enabled.
Document	Represents a file that a user has uploaded. Unlike Attachment objects, Documents are not attached to a parent object.
Event	Represents a calendar appointment event.
EventAttendee	Represents a person (User, Contact, or Lead) who has been invited to attend an Event, or a scheduled resource (such as a conference room) associated with the event.
Folder	Represents a repository for a Document, MailMergeTemplate, email template, or report. Only one type of item can be contained in a particular Folder.
Group	Represents a set of User.
GroupMember	Represents a User or Group that is a member of a public group.
Lead	Represents a lead, which is a prospect or potential Opportunity.
LeadStatus	Represents the status of a Lead, such as Open, Qualified, or Converted.
MailMergeTemplate	Represents a mail merge template (a Microsoft Word document) used for performing mail merges for your organization.
Note	Represents a note, which is text associated with an Attachment, Contact, Contract, Opportunity, or custom object.
Opportunity	Represents an opportunity, which is a sale or pending deal.
OpportunityCompetitor	Represents a competitor on an Opportunity.
OpportunityContactRole	Represents the association between an Opportunity and a Contact, with a specified Role name applied to the contact.
OpportunityHistory	Represents the history of an Opportunity.
OpportunityLineItem	Represents an opportunity line item, which is a member of the list of Product2s associated with an Opportunity, along with other information about those products on that opportunity.

Table 8: Supported Objects in the sforce API (Continued)

Object	Description
OpportunityLineItemSchedule	Represents information about the quantity, revenue distribution, and delivery dates for a particular OpportunityLineItem.
OpportunityShare	Represents a sharing entry on an Opportunity.
OpportunityStage	Represents the stage of an Opportunity in the sales pipeline, such as New Lead, Negotiating, Pending, Closed, and so on.
OpportunityTeamMember	Represents an individual User on the sales team of a particular Opportunity.
Partner	Represents the association between two particular Accounts or between a particular Opportunity and an Account.
PartnerRole	Represents a role for an account Partner, such as consultant, supplier, and so on.
Pricebook [Deprecated]	Represents a price book that contains the list of Product [Deprecated]s that your organization sells.
Pricebook2	Represents a price book that contains the list of Product2s that your organization sells.
PricebookEntry	Represents a product entry (an association between a Pricebook2 and Product2) in a pricebook.
Product [Deprecated]	Represents a product that your organization sells. A product is member of the list of items in a Pricebook [Deprecated].
Product2	Represents a product that your organization sells. A product is member of the list of items in a Pricebook2.
Profile	Represents a profile, which defines a set of permissions to perform different operations, such as querying, adding, updating, or deleting information.
RecordType	Represents a record type.
Scontrol	Represents an sforce control, which is custom content that is hosted by the server but executed by client applications.
Solution	Represents a solution, which is a detailed description of a customer issue and the resolution of that issue.
SolutionStatus	Represents the status of a Solution, such as Draft, Reviewed, and so on.
Task	Represents a task.
TaskPriority	Represents the priority (importance) of a Task, such as High, Normal, or Low.
TaskStatus	Represents the status of a Task, such as Not started, Completed, or Closed.
User	Represents a user in your organization.
UserRole	Represents a role in your organization.

Table 8: Supported Objects in the sforce API (Continued)

Object	Description
UserTeamMember	Represents a single User on the default sales team of another user.
WebLink	Represents a web link to an URL or Scontrol.

See Also
Concepts on page 101

ALLOWED API CALLS ON SFORCE OBJECTS

This topic describes the allowed API calls for each sforce object in the sforce Web services API. Note that *all* sforce objects support three calls: describeSObject, query, and retrieve.

sforce Object	create	update	delete	search	getDeleted getUpdated
Account	X	X	X	X	X
AccountContactRole	X	X	X		X
AccountShare	X	X	X		
AccountTeamMember	X	X	X	X	X
Approval	X	X	X	X	X
Asset	X	X	X	X	X
AssignmentRule					
Attachment	X	X	X	X	X
BusinessProcess	X	X			
Campaign	X	X	X	X	X
CampaignMember	X		X		X
Case	X	X	X	X	X
CaseComment	X	X			X
CaseHistory					X
CaseSolution	X		X		X
CaseStatus					
Contact	X	X	X	X	X
Contract	X	X	X	X	X
ContractContactRole	X	X	X		X
ContractStatus					
CurrencyType	X	X	X	X	X
Document	X	X	X		X

sforce Object	create	update	delete	search	getDeleted getUpdated
Event	X	X	X	X	X
EventAttendee					
Folder	X	X	X		
Group	X	X	X		X
GroupMember	X		X		X
Lead	X	X	X	X	X
LeadStatus					
MailMergeTemplate	X	X	X		
Note	X	X	X	X	X
Opportunity	X	X	X	X	X
OpportunityCompetitor	X	X	X		X
OpportunityContactRole	X	X	X		X
OpportunityHistory					
OpportunityLineItem	X	X	X		X
OpportunityLineItemSchedule	X	X	X		X
OpportunityShare	X	X	X		
OpportunityStage					
OpportunityTeamMember	X	X	X		X
Partner	X		X		X
PartnerRole					
Pricebook [Deprecated]	X	X			X
Pricebook2	X	X			X
PricebookEntry	X	X			X
Product [Deprecated]	X	X			X
Product2	X	X			X
Profile	X	X	X		
RecordType	X	X			
Scontrol	X	X	X		
Solution	X	X	X		X
SolutionStatus					
Task	X	X	X	X	X
TaskPriority					

sforce Object	create	update	delete	search	getDeleted getUpdated
TaskStatus					
User	X	X		X	X
UserRole	X	X	X	X	X
UserTeamMember	X	X	X		X
WebLink	X	X	X	X	X

Account

Represents an individual account, which is an organization involved with your business (such as customers, competitors, and partners).

Supported API Calls

create, update, delete, query, search, getDeleted, getUpdated, retrieve, describeSObject

Fields

For a complete list of fields in this object, see the Enterprise WSDL file for your organization and the salesforce.com online help.

Usage

Use Account objects to query and manage accounts in your organization. Client applications can create, update, delete, and query Attachments associated with an account via the sforce API.

Client applications can also create or update Account objects by converting a Lead via the convertLead call..

See Also

AccountShare on page 116
AccountTeamMember on page 118
Concepts on page 101

AccountContactRole

Represents the role that a given Contact plays on an Account.

Supported API Calls

create, update, delete, query, retrieve, getDeleted, getUpdated, describeSObject

Fields

For a complete list of fields in this object, see the Enterprise WSDL file for your organization and the salesforce.com online help.

Table 9: AccountContactRole Fields

Field	Data Type	Description
`AccountId`	ID	ID of the Account.
`ContactId`	ID	ID of the Contact associated with this account.
`IsPrimary`	boolean	Specifies whether this Contact plays the primary role on this Account (`true`) or not (`false`). Note that each account has only one primary contact role.
`Role`	string	Name of the role played by the Contact on this Account, such as Decision Maker, Approver, Buyer, and so on. Must be unique—there cannot be multiple records in which the `AccountId`, `ContactId`, and `Role` values are identical. Different contacts can play the same role on the same account. A contact can play different roles on the same account.

Usage

Use the AccountContactRole object to define the role that a given Contact plays on a given Account within the context of a specific Opportunity.

See Also

Account on page 115
Contact on page 130
Concepts on page 101

AccountShare

Represents a sharing entry on an Account.

Supported API Calls

create, update, delete, query, retrieve, getDeleted, getUpdated, describeSObject

Fields

For a complete list of fields in this object, see the Enterprise WSDL file for your organization and the salesforce.com online help.

Table 10: AccountShare Fields

Field	Data Type	Description
`AccountId`	ID	ID of the Account associated with this sharing entry. This field cannot be updated.
`UserOrGroupId`	ID	ID of the User or Group that has been given access to the Account. This field cannot be updated.

Table 10: AccountShare Fields (Continued)

Field	Data Type	Description
`AccountAccessLevel`	string	Level of access that the User or Group has to the Account. One of the following values: • **None** - User or Group cannot access the Account. • **Read** - User or Group can only view the Account. • **Edit** - User or Group can view or edit the Account. • **All** - User or Group can view, edit, delete, and share the Account with other Users. This value is not valid for create or update calls. This field must be set to an access level that is at least equal to the organization's default Account access level. In addition, either this field, the `OpportunityAccessLevel` field, or the `CaseAccessLevel` field must be set higher than the organization's default access level.
`CaseAccessLevel`	string	Level of access that the User or Group has to cases associated with the Account. One of the following values: • **None** - User or Group cannot access the Case. • **Read** - User or Group can only view the Case. • **Edit** - User or Group can view or edit the Case. This field must be set to an access level that is at least equal to the organization's default `caseAccessLevel`. This field cannot be updated via the API if the `AccountAccessLevel` field is set to "All." Using the sforce API, you cannot update this field for the associated Account owner. You must update the Account owner's `caseAccessLevel` via the salesforce.com user interface.
`OpportunityAccessLevel`	string	Level of access that the User or Group has to opportunities associated with the Account. One of the following values: • **None** - User or Group cannot access the associated opportunities. • **Read** - User or Group can only view the associated opportunities. • **Edit** - User or Group can view or edit the associated opportunities. This field must be set to an access level that is at least equal to the organization's default opportunityAccessLevel. This field cannot be updated via the API if the `AccountAccessLevel` field is set to "All." Using the sforce API, you cannot update this field for the associated Account owner. You must update the Account owner's opportunityAccessLevel via the salesforce.com user interface.

Table 10: AccountShare Fields (Continued)

Field	Data Type	Description
`RowCause`	string	Reason that this sharing entry exists. Read-only. One of the following values: • **Owner**—The User is the owner of the Account or is in a Role above the Account owner in the role hierarchy. • **Manual**—The User or Group has access because a User with "All" access manually shared the Account with them. • **Rule**—The User or Group has access via an Account sharing rule. • **ImplicitParent**—The User or Group has separate access to an Opportunity associated with this Account, and so they are automatically given "Read" access to the Account. • **Team**—The User or Group has team access (is an AccountTeamMember).

Usage

The AccountShare object allows you to determine which users and groups can view and/or edit Accounts owned by other users. For more information, see Sharing on page 21.

If you attempt to insert an AccountShare that matches an existing AccountShare record, the create call updates any modified fields and returns the existing record.

See Also

Account on page 115
Group on page 140
OpportunityShare on page 155
Concepts on page 101

AccountTeamMember

Represents a User who is a member of an Account team.

Supported API Calls

create, update, delete, query, search, getDeleted, getUpdated, retrieve, describeSObject

Fields

For a complete list of fields in this object, see the Enterprise WSDL file for your organization and the salesforce.com online help.

Table 11: AccountTeamMember Fields

Field	Data Type	Description
`AccountId`	ID	ID of the Account to which this user is a team member. Must be a valid Account ID. Required.

Table 11: AccountTeamMember Fields (Continued)

Field	Data Type	Description
`UserId`	ID	ID of the User who is a member of this account team. Must be a valid User ID. Required.
`TeamMemberRole`	string	Role associated with this team member. One of the valid team member roles defined for your organization. Required.

Usage

Use the AccountTeamMember object to manage members of a particular Account and to specify roles for those users on that account. This object is available only for Enterprise Edition users who have enabled the account team preference.

See Also

Concepts on page 101

Approval

Represents an approval request for a Contract.

Supported API Calls

create, update, query, search, getDeleted, getUpdated, retrieve, describeSObject

Fields

For a complete list of fields in this object, see the Enterprise WSDL file for your organization and the salesforce.com online help.

Table 12: Approval Fields

Field	Data Type	Description
`ApproveComment`	string	Text entered by the user when they approved *or* rejected this approval request. Required.
`OwnerId`	ID	ID of the User being asked to approve or reject the approval request. Must be a valid User ID. Required.
`ParentId`	ID	ID of the Contract associated with this approval request. Must be a valid contract ID. Required.
`RequestComment`	string	Text entered by the User who created the approval request. Optional. This field cannot be updated after the Approval has been created.

Table 12: Approval Fields (Continued)

Field	Data Type	Description
Status	string	Status of this approval request. One of the following picklist values: • Pending—Specified only when the Approval request is created (create call) • Approved—Specified only when the Approval request is approved (update call) • Rejected—Specified when the Approval request is rejected (update call) or when it is created (create call) and immediately rejected for archival/historical purposes.

Usage

The Approval object allows client applications to programmatically handle approval requests for a Contract. Initially, to request a Contract approval, a client application might create a new Approval request object, specifying the ContractId, **OwnerId** (user approving or rejecting the request), **Status** (Pending), and (optionally) **RequestComment** fields. Note that when a client application creates the first approval request, if the value of the Contract **status** field is Draft, then the **status** is automatically changed to **InApproval** (see ContractStatus on page 133 for more information).

A client application might subsequently update an existing Approval request, specifying the **Status** (Approved or Rejected) and an **ApproveComment** (required); the **RequestComment** field cannot be updated. Updating an Approval record (either to approve or reject) requires the client application to be logged in with "Approve Contract" permission. To update an Approval request, its **Status** must be Pending—a client application cannot update an Approval that has already been Approved or Rejected. To re-submit an Approval request for a given Contract, a client application must create a new, separate Approval object and repeat the approval process.

Once a Contract has been approved (not rejected), the sforce Web service automatically updates the Contract's **lastApprovedDate** field. The sforce Web service does *not* update its **status** field (which remains **InApproval**). To activate an approved Contract, it must be activated explicitly. Client applications can activate a Contract by setting the value in its Status field to Activated, or users can activate a contract using the salesforce.com user interface.

A Contract can have multiple approval requests in various states (Pending, Approved, and Rejected). In addition, one User can have multiple approval requests associated with the same Contract.

Client applications cannot explicitly delete Approval objects. Approval objects are deleted automatically if the parent Contract is deleted.

See Also

Concepts on page 101

Asset

Represents an asset (such as product previously sold and installed) owned by an Account or Contact.

Supported API Calls

create, update, delete, query, search, getDeleted, getUpdated, retrieve, describeSObject

Fields

For a complete list of fields in this object, see the Enterprise WSDL file for your organization and the salesforce.com online help.

Table 13: Asset Fields

Field	Data Type	Description
AccountId	ID	ID of the Account associated with this Asset. Must be a valid Account ID. Required if **ContactId** is not specified.
ContactId	ID	ID of the Contact associated with this asset. Must be a valid Contact ID that has an Account parent (but does not need to match the asset's AccountId). Required if **AccountId** is not specified.
Description	string	Description of this asset.
InstallDate	date	Date on which this asset was installed.
IsCompetitorProduct	boolean	Indicates whether this Asset represents a product sold by a competitor (True) or not (False).
Name	string	Name of the asset. Required.
Price	double	Price paid for this asset.
Product2Id	ID	ID of the Product2 associated with this asset. Must be a valid Product2 ID. Optional.
PurchaseDate	date	Date on which this asset was purchased.
Quantity	double	Quantity purchased / installed.
SerialNumber	string	Serial number for this asset.
Status	string	Customizable picklist of values. The default picklist includes the following values: • Purchased • Shipped • Installed • Registered • Obsolete
UsageEndDate	date	Date when usage for this asset ends / expires.

Usage

Use the Asset object to track assets previously sold into customer accounts. With asset tracking, a client application can quickly determine which products were previously sold and/or are currently installed at a specific account.

For example, your organization might want to renewal and up-sell opportunities on products sold in the past. Similarly, your organization might want to track competitive products that exist in a customer environment that could potentially be replaced or swapped out.

Asset tracking is also useful for product support, providing detailed information to assist with product-specific support issues. For example, the **PurchaseDate** or **SerialNumber** could indicate whether a given product has certain maintenance requirements, including product recalls.

Similarly, the **UsageEndDate** might indicate when the asset was removed from service or when a license or warranty expires.

If an application creates a new Asset, it must minimally specify its **Name** and either an **accountId**, **contactId**, or both.

See Also

Concepts on page 101

AssignmentRule

Represents an assignment rule associated with a Case or Lead.

Supported API Calls

query, retrieve, describeSObject

Fields

For a complete list of fields in this object, see the Enterprise WSDL file for your organization and the salesforce.com online help.

Table 14: AssignmentRule Fields

Field	Data Type	Description
Active	boolean	Indicates whether this assignment rule is active (`true`) or not (`false`).
Name	string	Name of this assignment rule.
RuleType	string	Type of assignment rule—`caseAssignment` or `leadAssignment`.

Usage

Before creating or updating a new Case or Lead, a client application can query (by name) the AssignmentRule to obtain the ID of the assignment rule to use, and then assigned that Id to the **assignmentRuleId** field of the AssignmentRuleHeader. For more information, see AssignmentRuleHeader on page 196.

Note that the AssignmentRule object is a read-only object. Assignment rules are created, configured, and deleted in the salesforce.com user interface.

See Also

Concepts on page 101

Attachment

Represents a file that a User has uploaded and attached to a parent object.

Supported API Calls

create, update, delete, query, search, getDeleted, getUpdated, retrieve, describeSObject

Fields

For a complete list of fields in this object, see the Enterprise WSDL file for your organization and the salesforce.com online help.

Table 15: Attachment Fields

Field	Data Type	Description
`Name`	string	Name of the attached file.
`ParentId`	ID	ID of the parent object of the Attachment. The following objects are supported as parents of Attachments: • Account • Campaign • Case • Contact • Contract • Opportunity • Solution
`Body`	base64Binary	Encoded file data.
`BodyLength`	int	Size of the file (in bytes).
`Private`	boolean	Indicates whether the Attachment is viewable only by the owner and administrators. If the private field is set to True for an Attachment, the Attachment can be viewed only by the Attachment owner and administrators. During a create or update call, it is possible to mark an Attachment as "private" even if you are not the Attachment owner. This can result in a situation in which you can no longer access the Attachment that you just inserted or updated.

All of the Attachment fields are accessible in the describeSObject and query calls. Using the create call, you can insert the `Name`, `ParentId`, `Body`, `Private`, and `OwnerId` fields. For modifying Attachments, the update call gives you access to change the `Name`, `Body`, `Private`, and `OwnerId` fields.

You can access all of the Attachment fields in a query call. However, you cannot receive the `Body` field for multiple Attachments in a single query call. If your query returns the `Body` field, your client application must ensure that only one row with one Attachment is returned; otherwise, an error occurs. A more effective approach is to return IDs (but not Attachments in the `Body` field) from a query call and then pass them into retrieve calls that return the `Body` field.

Usage

The sforce API sends and receives the binary file attachment data encoded as a base64Binary data type. Prior to create, clients must encode the binary attachment data as base64. Upon receiving an API response, clients must decode the base64 data to binary (this conversion is usually handled for you by the SOAP client).

The create call restricts Attachments to a maximum size of 5MB. For a file attached to a Solution, the limit is 1.5MB. The maximum email attachment size is 3MB.

Note that the API supports attachments on emails in create, update, and delete calls. The query call does not return attachments parented by emails, unless the user performing the query has the "Modify All Data" permission.

Note The search call does *not* search Attachment objects during text searches.

See Also

Concepts on page 101

BusinessProcess

Represents a business process.

Supported API Calls

create, update, query, retrieve, describeSObject

Fields

For a complete list of fields in this object, see the Enterprise WSDL file for your organization and the salesforce.com online help.

Table 16: Business Process Fields

Field	Data Type	Description
`Name`	string	Name of this business process. Up to 80 characters.
`Description`	string	Description of this business process. Up to 255 characters.
`IsActive`	boolean	Indicates whether this BusinessProcess can be presented to users in the salesforce.com user interface (True) or not (False) when creating a new record type or changing the business process of an existing record type.

Usage

Use the BusinessProcess object to offer different subsets of picklist values to different users for the `Lead Status`, `Case Status`, and `Opportunity Stage` fields (see the LeadStatus, CaseStatus, and OpportunityStage objects, respectively). Similar to a RecordType, a BusinessProcess identifies the type of a row in a Case, Lead, or Opportunity and implies a subset of picklist values for these three fields. The values for the remaining picklist fields are driven off of RecordType.

See Also

Concepts on page 101

Campaign

Represents and tracks a marketing campaign, such as a direct mail promotion, webinar, or trade show.

Supported API Calls

create, update, delete, query, search, getDeleted, getUpdated, retrieve, describeSObject

Fields

The Campaign statistics fields are read-only (as in the salesforce.com user interface). You cannot update the statistics via the sforce API. For a complete list of fields in this object, see the Enterprise WSDL file for your organization and the salesforce.com online help.

Usage

Client applications can create, update, delete, and query Attachments associated with a campaign via the sforce API.

The Campaign object is defined only for those organizations that have the Marketing feature enabled and valid Marketing licenses. In addition, it is accessible only to those users that are enabled as Marketing Users. If the organization does not have the Marketing feature or valid Marketing licenses, the Campaign object does not appear in the describeGlobal call, and you cannot use describeSObject or query with the Campaign object.

Note The main constituents of campaigns are CampaignMember. You will commonly need to update campaigns with CampaignMember. See CampaignMember on page 125.

See Also

Concepts on page 101

CampaignMember

Represents the association between a Campaign and either a Lead or Contact.

Supported API Calls

create, update, delete, query, getDeleted, getUpdated, retrieve, describeSObject

Fields

For a complete list of fields in this object, see the Enterprise WSDL file for your organization and the salesforce.com online help.

Status field

The **Status** field is a picklist that directly controls the **Responded** flag on a CampaignMember. You cannot directly set the **Responded** flag, as it is read-only, but you can set it indirectly by setting the **Status** field in a create or update call. Each predefined **Status** field value implies a **Responded** flag value. Each time you update the **Status** field, you implicitly update the **Responded** flag on the CampaignMember.

In the salesforce.com user interface, Marketing Users can define the valid CampaignMember status values for the **Status** picklist. They can choose one status as the "default" CampaignMember status. For each **Status** field value, they can also select which values should be counted as "Responded," meaning that the **Responded** flag will be set to True for those **Status** values.

Usage

Each CampaignMember record has a unique ID. Each individual CampaignMember record must contain either a contactId or a leadId, but cannot contain both. Any attempt to create a single CampaignMember with both a contactId and a leadId results in an error. However, you can create separate CampaignMember records on a Campaign, one for the Lead and one for the Contact.

The CampaignMember object is defined only for those organizations that have the Marketing feature and valid Marketing licenses. In addition, the object is accessible only to those users that are enabled as Marketing Users. If the organization does not have the Marketing feature or valid Marketing licenses, the CampaignMember object does not appear in the describeGlobal call, and you cannot use describeSObject or query with the CampaignMember object.

Inserting, Updating, and Deleting Campaign Members

You can indirectly update CampaignMembers by sending a create request. A create call for CampaignMembers is interpreted as an auto-insert-or-update call. The sforce API automatically determines whether a CampaignMember exists with the specified Campaign Id and Contact or Lead Id. If the CampaignMember does not exist for the given contact or lead Id, then a create is performed. If the CampaignMember already exists, the call is interpreted as an update and the **Status** field and **Responded** flag on the existing record are updated. Thus, you cannot create duplicate CampaignMember records, since any attempt to create a duplicate record simply updates the existing record.

During a create or update call, the sforce Web service verifies whether the **Status** field value specified in the call is a valid CampaignMember status for the given Campaign. If the specified **Status** value is a valid CampaignMember status, the sforce Web service assigns that value to the CampaignMember **Status** field and updates the **Responded** flag with the associated value. If the specified **Status** value is not a valid CampaignMember status, the API assigns the default CampaignMember status to the **Status** field and updates the **Responded** flag with the associated value. However, if the given Campaign does not have a default CampaignMember status, the API assigns the value specified in the call to the **Status** field, and the **Responded** flag is set to False.

See Also

Concepts on page 101

Case

Represents a case, which is a customer issue such as a customer's feedback, problem, or question.

Supported API Calls

create, update, delete, query, search, getDeleted, getUpdated, retrieve, describeSObject

Fields

For a complete list of fields in this object, see the Enterprise WSDL file for your organization and the salesforce.com online help.

Case Number Field

The **CaseNumber** field is assigned automatically when each Case is inserted. It cannot be set directly, and it cannot be modified after the Case is created.

Create-Only Fields

Cases have several fields that can only be set when calling create. They cannot be updated after the Case has been created. Those fields are **Name**, **Email**, **Phone**, and **Company**.

Status Field and IsClosed Flag

The **Status** field is a picklist (see the CaseStatus object) that also directly controls the **IsClosed** flag. You cannot directly set the **IsClosed** flag, but you can set it indirectly by setting the **Status** field. Each predefined **Status** field value implies an **IsClosed** flag value.

IsEscalated Flag

Cases can have a special status called "escalated." (See the salesforce.com online help documentation for more information on Case escalation.) A Case's escalated state does not affect how you can use a Case, or whether you can query, update, or delete it. However, you cannot set the **IsEscalated** flag via the sforce API.

IsVisibleInCss Flag

The **IsVisibleInCss** flag controls whether a case can viewed in the CSS portal.

Usage

Use the Case object to manage cases for your organization. Client applications can create, update, delete, and query Attachments associated with a case via the sforce API.

When you create or update a case, your client application can have the case automatically assigned to one or more Users based on assignment rules that have been configured in the salesforce.com user interface. To use this feature, your client application needs to set *either* of the following options (but not both) in the AssignmentRuleHeader used in the create or update call (note that SaveOptions (Deprecated) is deprecated):

Table 17: AssignmentRuleHeader for Automatic Rule-Based Case Assignment

Field	Data Type	Description
assignmentRuleId	ID	ID of the assignment rule to use. Can be an inactive assignment rule. If unspecified and **useDefaultRule** is True, then the default assignment rule is used. To find the ID for a given assignment rule, you query the AssignmentRule object (specifying `RuleType="caseAssignment"`), iterate through the returned AssignmentRule objects, find the one you want to use, retrieve its ID, and then specify its ID in this field in the AssignmentRuleHeader.
useDefaultRule	boolean	Specifies whether to use the default rule for rule-based assignment (True) or not (False). The default rule is assigned by users in the salesforce.com user interface.

For a code example that shows setting the AssignmentRuleHeader for a Lead (which is similar to setting the AssignmentRuleHeader for a Case), see Lead on page 141.

See Also

Concepts on page 101

CaseComment

Represents a comment that provides additional information about the associated Case.

Supported API Calls

create, update, query, search, getDeleted, getUpdated, retrieve, describeSObject

Fields

For a complete list of fields in this object, see the Enterprise WSDL file for your organization and the salesforce.com online help.

Table 18: CaseComment Fields

Field	Data Type	Description
`ParentId`	ID	ID of the parent Case of the CaseComment.
`Published`	boolean	Indicates whether the CaseComment is visible to customers in the Self-Service portal. This is the only field that can be updated via the sforce API. All of the other case CaseComment cannot be updated.
`CommentBody`	string	Text of the CaseComment. The maximum size of the comment body is 4000 bytes.

Usage

In the salesforce.com user interface, comments are generally entered by users working on a particular Case. All users have access to create and view CaseComments in the salesforce.com user interface and when using the sforce API. In both the salesforce.com user interface and via the sforce API, CaseComments cannot be modified after insertion, except to update the `Published` field. You cannot delete CaseComments by any means.

See Also

Concepts on page 101

CaseHistory

Represents historical information about changes that have been made to the associated Case.

Supported API Calls

query, retrieve, describeSObject

Fields

For a complete list of fields in this object, see the Enterprise WSDL file for your organization and the salesforce.com online help.

Table 19: CaseHistory Fields

Field	Data Type	Description
`CaseId`	ID	ID of the Case associated with the CaseHistory entry.

Table 19: CaseHistory Fields (Continued)

Field	Data Type	Description
`Field`	string	Name of the case field that was modified, or a special value to indicate some other modification to the case. The possible values, in addition to the case field names, are: • **ownerAssignment** - The owner of the case was changed. • **ownerAccepted** - A User took ownership of a case from a queue. • **ownerEscalated** - The owner of the case was changed due to case escalation. • **external** - A User made the case visible to customers in the Customer Self-Service Portal.
`OldValue`	string	Previous value of the modified case field.
`NewValue`	string	New value of the modified case field.

Usage

The CaseHistory object is always read-only in salesforce.com. Case history entries are indirectly created by modifying a case via the salesforce.com user interface or the sforce API.

See Also

Concepts on page 101

CaseSolution

Represents the association between a particular Case and a particular Solution.

Supported API Calls

create, delete, query, getDeleted, getUpdated, retrieve, describeSObject

Fields

For a complete list of fields in this object, see the Enterprise WSDL file for your organization and the salesforce.com online help.

Table 20: CaseSolution Fields

Field	Data Type	Description
`CaseId`	ID	ID of the Case associated with the Solution.
`SolutionId`	ID	ID of the Solution associated with the case.

Usage

You cannot update CaseSolutions via the sforce API. If you attempt to insert a CaseSolution that matches an existing CaseSolution record, the create call simply returns the existing record.

See Also

Concepts on page 101

CaseStatus

Represents the status of a Case, such as New, On Hold, In Process, and so on.

Supported API Calls

query, retrieve, describeSObject

Fields

For a complete list of fields in this object, see the Enterprise WSDL file for your organization and the salesforce.com online help.

Table 21: CaseStatus Fields

Field	Data Type	Description
`IsClosed`	boolean	Indicates whether this case status value represents a closed Case (True) or not (False). Multiple case status values can represent a closed Case.
`IsDefault`	boolean	Indicates whether this is the default case status value (True) or not (False) in the picklist.
`MasterLabel`	string	Master label for this case status value. This display value is the internal label that does not get translated.
`SortOrder`	int	Number used to sort this value in the case status picklist. These numbers are not guaranteed to be sequential, as some previous case status values might have been deleted.

Usage

The CaseStatus object represents a value in the case status picklist. The case status picklist provides additional information about the status of a Case, such as whether a given status value represents an open or closed case. Your client application can invoke the query call on the CaseStatus object to retrieve the set of values in the case status picklist, and then use that information while processing Case objects to determine more information about a given case. For example, the application could test whether a given case is open or closed based on its `Status` value and the value of the `IsClosed` property in the associated CaseStatus object.

The CaseStatus object is read-only via the sforce API. With sufficient permissions, your client application can invoke the query and describeSObject calls on CaseStatus objects.

See Also

Concepts on page 101

Contact

Represents a contact, which is an individual associated with your Accounts.

Supported API Calls

create, update, delete, query, search, getDeleted, getUpdated, retrieve, describeSObject

Fields

For a complete list of fields in this object, see the Enterprise WSDL file for your organization and the salesforce.com online help.

Usage

Use the Contact object to manage individuals who are associated with Accounts in your organization. Client applications can create, update, delete, and query Attachments associated with a contact via the sforce API.

Client applications can also create or update Contact objects by converting a Lead via the convertLead call..

See Also

Concepts on page 101

Contract

Represents a contract (a business agreement) associated with an Account.

Supported API Calls

create, update, delete, query, search, getDeleted, getUpdated, retrieve, describeSObject

Fields

For a complete list of fields in this object, see the Enterprise WSDL file for your organization and the salesforce.com online help.

Table 22: Contract Fields

Field	Data Type	Description
`AccountId`	ID	ID of the Account associated with this contract.
`ActivatedById`	ID	ID of the User who activated this contract.
`ActivatedDate`	dateTime	Date/time when this contract was activated.
`CompanySignedDate`	date	Date on which this contract was signed by your organization.
`CompanySignedId`	ID	ID of the User who signed this contract.
`ContractNumber`	string	Number of this contract.
`ContractTerm`	integer	Number of months that this contract is valid.
`CustomerSignedDate`	date	Date on which the customer signed this contract.
`CustomerSignedId`	ID	ID of the Contact who signed this contract.

Table 22: Contract Fields (Continued)

Field	Data Type	Description
EndDate	date	Calculated end date of this contract. Read-only. This value is calculated by adding the ContractTerm to the StartDate.
OwnerExpirationNotice	string	Number of days ahead of the contract end date (15, 30, 45, 60, 90, and 120). Used to notify the owner in advance that the contract is ending.
StartDate	date	Start date for this contract.
Status	string	Status for the contract. See ContractStatus on page 133 for a list of valid status codes.

Usage

The Contract object represents a business agreement. Client applications can create, update, query, and retrieve contracts via the sforce API.

The Status field specifies the current state of a contract. Status strings (defined in the ContractStatus object) represent its current state (Draft, InApproval, or Activated).

Client applications must initially create a Contract in a non-Activated state. Client applications can subsequently activate a Contract by calling update and setting the value in its Status field to Activated; however, the Status field is the *only* field that the client application can set in the update call when activating the Contract.

Once a Contract has been activated, your client application cannot change its status; however, prior to activation, your client application *can* change the status from Draft to InApproval via the sforce API. Also, your client application can delete contracts whose status is Draft or InApproval but *not* when a contract is Activated.

Client applications can create, update, delete, and query Attachments associated with a contract via the sforce API.

See Also

Concepts on page 101

ContractContactRole

Represents the role that a given Contact plays on a Contract.

Supported API Calls

create, update, delete, query, retrieve, getDeleted, getUpdated, describeSObject

Fields

For a complete list of fields in this object, see the Enterprise WSDL file for your organization and the salesforce.com online help.

Table 23: ContractContactRole Fields

Field	Data Type	Description
ContactId	ID	ID of the Contact associated with this Contract.

Table 23: ContractContactRole Fields (Continued)

Field	Data Type	Description
`ContractId`	ID	ID of the Contract.
`IsPrimary`	boolean	Specifies whether this Contact plays the primary role on this Contract (`true`) or not (`false`). Note that each contract has only one primary contact role.
`Role`	string	Name of the role played by the Contact on this Contract, such as Decision Maker, Approver, Buyer, and so on. Must be unique—there cannot be multiple records in which the `ContractId`, `ContactId`, and `Role` values are identical. Different contacts can play the same role on the same contract. A contact can play different roles on the same contract.

Usage

Use the ContractContactRole object to define the role that a given Contact plays on a given Contract within the context of a specific Opportunity.

See Also

Contact on page 130
Contract on page 131
Concepts on page 101

ContractStatus

Represents the status of a Contract, such as Draft, InApproval, Activated, Terminated, or Expired.

Supported API Calls

query, retrieve, describeSObject

Fields

For a complete list of fields in this object, see the Enterprise WSDL file for your organization and the salesforce.com online help.

Table 24: ContractStatus Fields

Field	Data Type	Description
`IsDefault`	boolean	Indicates whether this is the default contract status value (True) or not (False) in the picklist.
`MasterLabel`	string	Master label for this contract status value. This display value is the internal label that does not get translated.

Table 24: ContractStatus Fields (Continued)

Field	Data Type	Description
`SortOrder`	int	Number used to sort this value in the contract status picklist. These numbers are not guaranteed to be sequential, as some previous contract status values might have been deleted.
`StatusCode`	string	Code indicating the status of a contract. One of the following values: • `Draft` • `InApproval` • `Activated` Two other values (`Terminated` and `Expired`) are defined but are not available for use via the sforce API.

Usage

The ContractStatus object represents a value in the contract status picklist. The contract status picklist provides additional information about the status of a Contract, such as its current state (`Draft`, `InApproval`, or `Activated`). Your client application can invoke the query call on the ContractStatus object to retrieve the set of values in the contract status picklist, and then use that information while processing Contract objects to determine more information about a given contract. For example, the application could test whether a given contract is activated based on its `Status` value and the value of the `StatusCode` property in the associated ContractStatus object.

The ContractStatus object is read-only via the sforce API. With sufficient permissions, your client application can invoke the query and describeSObject calls on ContractStatus objects.

See Also

Concepts on page 101

CurrencyType

Represents the currencies used by an organization for which the multi-currency feature is enabled.

Supported API Calls

create, update, query, search, getUpdated, retrieve, describeSObject

Fields

For a complete list of fields in this object, see the Enterprise WSDL file for your organization and the salesforce.com online help.

Table 25: CurrencyType Fields

Field	Data Type	Description
`IsoCode`	string	ISO code of the currency. Required. Must be one of the valid alphabetic, three-letter currency ISO code defined by the ISO 4217 standard, such as USD, GBP, or JPY. Must be unique within your organization.
`ConversionRate`	double	Conversion rate of this currency type against the corporate currency.
`DecimalPlaces`	int	For this currency, specifies the number of digits to the right of the decimal point, such as zero (0) for JPY or 2 for USD. Required.
`IsActive`	boolean	Indicates whether this currency type is active (True) or not (False). Inactive currency types do not appear in picklists in the salesforce.com user interface.
`IsCorporate`	boolean	Indicates whether this currency type is the corporate currency (True) or not (False). Required. All other currency conversion rates are applied against this corporate currency. If a currency is already defined as the corporate currency in the salesforce.com user interface, it cannot be unset. When a non-corporate currency is set to a corporate currency, the system will reconfigure all conversion rates based on the new corporate currency.

Usage

For multi-currency organizations only, use the CurrencyType object to define the currencies that your organization uses. This object is not available in single-currency organizations.

Your client application cannot delete a CurrencyType object. In addition, you need "Customize salesforce.com" permission to edit a CurrencyType.

See Also

Concepts on page 101

Document

Represents a file that a user has uploaded. Unlike Attachment objects, Documents are not attached to a parent object.

Supported API Calls

create, update, delete, query, getDeleted, getUpdated, retrieve, describeSObject

Fields

For a complete list of fields in this object, see the Enterprise WSDL file for your organization and the salesforce.com online help.

Table 26: Document Fields

Field	Data Type	Description
`FolderId`	ID	ID of the Folder that contains the Document. See Folder on page 139.
`Name`	string	Name of the Document.
`Type`	string	File type of the Document. In general, the values match the file extension for the type of Document (such as `pdf` or `jpg`.).
`Body`	base64Binary	Encoded file data. If specified, then do not specify an URL.
`BodyLength`	int	Size of the file (in bytes).
`URL`	string	URL reference to the file (instead of storing it in the database). If specified, do not specify the Body / BodyLength.
`Description`	string	Text description of the Document.
`Author`	string	ID of the User who is responsible for the Document.

Usage

You must have the "Edit" permission on Documents and the appropriate access to the Folder that contains a document in order to create or update a Document in that Folder.

When calling create or update for a document, a client application can specify a value in *either* the `Body` or `URL` fields—but not both.

Encoded Data

The sforce API sends and receives the binary file data encoded as a base64Binary data type. Prior to create, clients must encode the binary file data as base64. Upon receiving an API response, clients must decode the base64 data to binary (this conversion is usually handled for you by the SOAP client).

Maximum Document Size

The create and update calls restrict documents to a maximum size of 5MB.

See Also

Concepts on page 101

Event

Represents a calendar appointment event.

Supported API Calls

create, update, delete, query, search, getDeleted, getUpdated, retrieve, describeSObject

Fields

For a complete list of fields in this object, see the Enterprise WSDL file for your organization and the salesforce.com online help.

WhoId and WhatId Fields

The Event object has **WhoId** and **WhatId** cross-reference ID fields. These are cross-reference fields that can each point to one of several other objects. The **WhoId** field can point to a Contact or Lead, and the **WhatId** field can point to an Account, Opportunity, Campaign, Case, or a custom object. In addition, if the **WhoId** field refers to a Lead, then the **WhatId** field must be empty.

ActivityDateTime and ActivityDate Fields

The salesforce.com user interface has a single **Due Date** field for Events. However in the sforce API, the due date information is contained in either the **ActivityDateTime** field or the **ActivityDate** field, depending on the value of the Event **IsAllDayEvent** flag.

- **ActivityDateTime** - If the Event **IsAllDayEvent** flag is set to False (indicating that it is not an all day Event), then the Event due date information is contained in the **ActivityDateTime** field. This field is a regular Date/Time field with a relevant time portion. The time portion is always transferred in the GMT/UTC time zone. You need to translate the time portion to or from a local time zone for the user or the application, as appropriate. For more information, see DateTime Field Type on page 103.
- **ActivityDate** - If the Event **IsAllDayEvent** flag is set to True (indicating that it is an all day Event), then the Event due date information is contained in the **ActivityDate** field. This field is a date field with a timestamp that is always set to midnight in the GMT/UTC time zone. The timestamp is not relevant, and you should not attempt to alter it to account for any time zone differences. For more information, see Date Field Type on page 103.

When querying for events with a specific due date, you must filter on both the **ActivityDateTime** and **ActivityDate** fields. For example to find all events with a due date of February 14, 2003, you need two filters:

- one filter with the **ActivityDate** field equal to midnight GMT on February 14, 2003
- one filter with the **ActivityDateTime** field greater than or equal to midnight on February 14, 2003 in the user's local time zone AND less than or equal to midnight on February 15, 2003 in the user's local time zone

IsPrivate Field

The **IsPrivate** field indicates whether users other than the creator of this event can (False) or cannot (True) see the event details when viewing the event user's calendar. However, users with the "View All Data" or "Modify All Data" permission can see private events in reports and searches, or when viewing other users' calendars. Private events cannot be associated with opportunities, accounts, cases, campaigns, contracts, leads, or contacts.

ShowAs Field

The **showAs** field is a picklist that determines how this event appears when another user views the calendar: busy, out of office, or free time.

Usage

Use Events to manage calendar appointments.

Archived Activities

Sforce archives older events and Tasks according to the criteria listed below. In the salesforce.com user interface, users can view archived activities only in the **Printable View** or by clicking **View All** on the Activity History related list or by doing an advanced search. However in the sforce API, archived activities are not accessible.

Sforce archives activities according to the following criteria.

- Events with an **ActivityDateTime** or **ActivityDate** value greater than or equal to 365 days old
- Tasks with a **Closed** flag value of True and an **ActivityDate** value greater than or equal to 365 days old
- Tasks with a **Closed** flag value of True, a blank **ActivityDate** field, and a create date greater than or equal to 365 days ago

If you use the sforce API to insert activities that meet these criteria, the activities will be archived during the next run of the archival background process.

See Also

Concepts on page 101

EventAttendee

Represents a person (User, Contact, or Lead) who has been invited to attend an Event, or a scheduled resource (such as a conference room) associated with the event.

Supported API Calls

query, retrieve, describeSObject, getUpdated, getDeleted

Fields

For a complete list of fields in this object, see the Enterprise WSDL file for your organization and the salesforce.com online help.

Table 27: EventAttendee Fields

Field	Data Type	Description
AttendeeId	ID	ID of the person (User, Contact, or Lead) who has been invited to attend the Event, or the scheduled resource associated with this Event.
EventId	ID	ID of the Event.
RespondedDate	dateTime	Date / time when the attendee (invitee) responded.
Response	string	Optional text that the attendee entered when responding to the event request.
Status	string	Attendee status. One of the following values: • `New`—Invitee has received the invitation but has not yet responded. • `Declined`—Invitee has declined to attend the event. • `Accepted`—Invitee has accepted the invitation to attend the event. • `Deleted`—The invitation has been deleted. • `Uninvited`—Reserved for future use. • `Maybe`—Reserved for future use.

Usage

The EventAttendee object is a read-only object that provides information about who has been invited to attend a particular event and their response to that invitation. A client application can, for example, query the EventAttendee object for a given event, iterate through the list, examine the status, and send email notifications to every person who accepted the invitation.

To determine all the events that a particular person is attending during a given time period (for example, next week), a client application could query the Event object for a given date range, iterate through the results and, for each event, query the EventAttendee object and determine whether the particular person (`attendeeId`) has accepted an invitation to that event.

See Also

Concepts on page 101

Folder

Represents a repository for a Document, MailMergeTemplate, email template, or report. Only one type of item can be contained in a particular Folder.

Supported API Calls

create, update, delete, getDeleted, getUpdated, query, retrieve, describeSObject

Fields

For a complete list of fields in this object, see the Enterprise WSDL file for your organization and the salesforce.com online help.

Table 28: Folder Fields

Field	Data Type	Description
Name	string	Name of the Folder.
Type	string	Type of objects contained in the Folder. The **Type** field cannot be updated. One of the following values: • **Document** • **Email template** • **Report**
AccessType	string	Indicates who can access the Folder. One of the following values: • **Shared** - Folder is accessible only by Users in a particular Group or UserRole. The API does not allow you to view, insert, or update which group or Role the Folder is shared with. • **Public** - Folder is accessible by all users. • **Hidden** - Folder is hidden from everyone.
ReadOnly	boolean	Indicates whether you can add data to this Folder.

Usage

You must have the "Modify All Data" permission to create, update, and delete document folders, email template folders, or report folders. To query Folders, no special permissions are needed.

See Also

Concepts on page 101

Group

Represents a set of Users.

Supported API Calls

create, update, delete, query, getDeleted, getUpdated, retrieve, describeSObject

Fields

For a complete list of fields in this object, see the Enterprise WSDL file for your organization and the salesforce.com online help.

Table 29: Group Fields

Field	Data Type	Description
`Name`	string	Name of the Group.
`RelatedId`	ID	For Groups of type "Role," the ID of the associated UserRole. Read-only.
`Type`	string	Type of the Group. One of the following values: • **Regular**—Standard public Group. When you create a Group, its type must be "`Regular`". • **Role**—Public Group that includes all of the Users in a particular UserRole. • **RoleAndSubordinates**—Public Group that includes all of the User in a particular UserRole and all of the Users in UserRoles below that UserRole. • **Organization**—Public Group that includes all of the Users in the organization. This Group is read-only. • **Case**—Public group of people that comprise a queue that can own a Case. • **Lead** —Public group of people that comprise a queue that can own a Lead.
`Email`	string	Email address for a group of type `Case`. Applies only for a case queue.

Usage

Groups can be deleted. Any User can access the Group object—no special permissions are needed.

Only public Groups are accessible via the sforce API. Personal Groups are not available.

See Also

GroupMember on page 141
Concepts on page 101

GroupMember

Represents a User or Group that is a members of a public group.

Supported API Calls

create, delete, query, getDeleted, retrieve, describeSObject

Fields

For a complete list of fields in this object, see the Enterprise WSDL file for your organization and the salesforce.com online help.

Table 30: GroupMember Fields

Field	Data Type	Description
`GroupId`	ID	ID of the Group.
`UserOrGroupId`	ID	ID of the User or Group that is a direct member of the group.

Usage

GroupMembers cannot be updated. Any user can access the GroupMember object—no special permissions are needed.

A GroupMember record exists for every User or Group who is a direct member of a public group whose `Type` field is set to "Regular." Users who are indirect members of "Regular" public groups are not listed as group members. A User can be an indirect member of a group if he or she is in a UserRole above the direct group member in the hierarchy, or if he or she is a member of a group that is included as a subgroup in that group.

If you attempt to insert a GroupMember that matches an existing GroupMember record, the create call simply returns the existing record.

See Also

Group on page 140
Concepts on page 101

Lead

Represents a lead, which is a prospect or potential Opportunity.

Supported API Calls

convertLead, create, update, delete, query, search, getDeleted, getUpdated, retrieve, describeSObject

Fields

For a complete list of fields in this object, see the Enterprise WSDL file for your organization and the salesforce.com online help.

Converted Leads

Leads have a special state to indicate that they have been converted into an Account, Contact, and Opportunity. Your client application can convert leads via the convertLead call. Users can also convert Leads through the salesforce.com user interface. Once a Lead has been converted, it is read-only. You cannot update or delete a converted Lead. However, you can query converted Leads using the query call.

Leads have several fields that indicate their converted status. These special fields are read-only via the sforce API. You cannot set these fields directly; they are set when converting the Lead in the salesforce.com user interface. The fields are:

Table 31: Fields on Converted Leads

Field	Data Type	Description
`Status`	string	Status code for this lead. Status codes are defined in the lead status picklist and represented in the sforce API by the LeadStatus object.
`Converted`	boolean	Indicates whether the Lead has been converted (True) or not (False).
`ConvertedAccountId`	ID	Object reference ID that points to the Account into which the Lead has been converted.
`ConvertedContactId`	ID	Object reference ID that points to the Contact into which the Lead has been converted.
`ConvertedDate`	date	Date on which this Lead was converted.
`ConvertedOpportunityId`	ID	Object reference ID that points to the Opportunity into which the Lead has been converted.

Unread Leads

Leads have a special state to indicate that they have not been viewed or edited by the lead owner. In the salesforce.com user interface, this is helpful for users to know which leads have been assigned to them but which they have not touched yet. The `IsUnreadByOwner` field is True if the lead owner has not yet viewed or edited the lead, and False if the lead owner has viewed or edited the lead at least once.

Lead Status Picklist

Each `Lead Status` value corresponds to either a converted or unconverted status in the lead status picklist, as defined in the salesforce.com user interface. To obtain the lead status values in the picklist, a client application can invoke the query call on the LeadStatus object.

You cannot convert a lead via the API by changing the Status field to one of the "converted" lead status values. When you convert qualified leads into an account, contact, and opportunity, you can select one of the "converted" status types to assign to the lead. Leads with a "converted" status type are no longer available in the Leads tab, although you can include them in reports.

Usage

To update a Lead or call convertLead, your client application must be logged in with "Edit" permission on Leads.

When you create or update a lead, your client application can have the lead automatically assigned to one or more Users based on assignment rules that have been configured in the salesforce.com user interface. To use this feature, your client application needs to set *either* of the following options (but not both) in the create or update call (note that SaveOptions (Deprecated) is deprecated):

Table 32: AssignmentRuleHeader for Automatic Rule-Based Case Assignment

Field	Data Type	Description
assignmentRuleId	ID	ID of the assignment rule to use. Can be an inactive assignment rule. If unspecified and **useDefaultRule** is True, then the default assignment rule is used. To find the ID for a given assignment rule, you query the AssignmentRule object (specifying `RuleType="leadAssignment"`), iterate through the returned AssignmentRule objects, find the one you want to use, retrieve its ID, and then specify its ID in this field in the AssignmentRuleHeader.
useDefaultRule	boolean	Specifies whether to use the default rule for rule-based assignment (True) or not (False). The default rule is assigned by users in the salesforce.com user interface.

The following code example shows how to automatically assign a newly-created lead.

```
package com.sforce;
import java.net.MalformedURLException;
import java.net.URL;
import java.rmi.RemoteException;
import javax.xml.rpc.ServiceException;

import com.sforce.soap.enterprise.LoginResult;
import com.sforce.soap.enterprise.QueryResult;
import com.sforce.soap.enterprise.SaveResult;
import com.sforce.soap.enterprise.SforceServiceLocator;
import com.sforce.soap.enterprise.SoapBindingStub;
import com.sforce.soap.enterprise._AssignmentRuleHeader;
import com.sforce.soap.enterprise._SessionHeader;
import com.sforce.soap.enterprise.fault.LoginFault;
import com.sforce.soap.enterprise.fault.UnexpectedErrorFault;
import com.sforce.soap.enterprise.sobject.Lead;
import com.sforce.soap.enterprise.sobject.SObject;

public class LeadAssignment {

    static LeadAssignment _leadAssignment;

    public static void main(String[] args) {
        _leadAssignment = new LeadAssignment();
        try {
            _leadAssignment.CreateLead();
        } catch (Exception e) {
            e.printStackTrace();
        }
    }

    public void CreateLead() throws UnexpectedErrorFault, LoginFault,
RemoteException {
        //Create the proxy binding and login
        SoapBindingStub binding = (SoapBindingStub) new
```



```
SforceServiceLocator().getSoap();
        LoginResult lr = binding.login("user@domain.net", "secret");

        //Reset the binding to use the endpoint returned from login
        binding._setProperty(SoapBindingStub.ENDPOINT_ADDRESS_POINT,
loginResult.getServerUrl());

        //Create the session id header, and add it to the proxy binding
        _SessionHeader sh = new _SessionHeader();
        sh.setSessionId(lr.getSessionId());
        binding.setHeader(new
SforceServiceLocator().getServiceName().getNamespaceURI(), "SessionHeader", sh);

        //Create a new case and assign various properties
        Lead lead = new Lead();

        lead.setFirstName("Joe");
        lead.setLastName("Smith");
        lead.setCompany("ABC Corporation");
        lead.setLeadSource("API");
        //The lead assignment rule will assign any new leads that
        //have "API" as the LeadSource to a particular user

        //Create the assignment rule header and add it to the proxy binding
        _AssignmentRuleHeader arh = new _AssignmentRuleHeader();

        //In this sample we will look for a particular rule and if found
        //use the id for the lead assignment.  If it is not found we will
        //instruct the call to use the current default rule.  You cannot use
        //both of these values together.
       QueryResult qr = binding.query("Select Id From AssignmentRule where Name =
'Mass Mail Campaign' and RuleType = 'leadAssignment'");
        if (qr.getSize() == 0) {
            arh.setUseDefaultRule(new Boolean(true));
        } else {
            arh.setAssignmentRuleId(qr.getRecords(0).getId());
        }

        binding.setHeader(new
        SforceServiceLocator().getServiceName().getNamespaceURI(),
"AssignmentRuleHeader", arh);

        // Every operation that results in a new or updated case, will
        // use the specified rule until the header is removed from the
        // proxy binding.
        SaveResult[] sr = binding.create(new SObject[] {lead});
        for (int i=0;i<sr.length;i++) {
            if (sr[i].isSuccess())
                System.out.println("Successfully creaeted lead with id of: " +
sr[i].getId().getValue() + ".");
            else
                System.out.println("Error creating lead: " +
sr[i].getErrors(0).getMessage());
        }

        // This call effectively removes the header, the next lead will
        // be assigned to the default lead owner.  Remember to add the
        // session header back in.
        binding.clearHeaders();
        binding.setHeader(new
```



```
        SforceServiceLocator().getServiceName().getNamespaceURI(),
"SessionHeader", sh);

    }
}
```

See Also

Concepts on page 101

LeadStatus

Represents the status of a Lead, such as Open, Qualified, or Converted.

Supported API Calls

query, retrieve, describeSObject

Fields

For a complete list of fields in this object, see the Enterprise WSDL file for your organization and the salesforce.com online help.

Table 33: LeadStatus Fields

Field	Data Type	Description
IsConverted	boolean	Indicates whether this lead status value represents a converted lead (True) or not (False). Multiple lead status values can represent a converted lead. For more information, see convertLead on page 43.
IsDefault	boolean	Indicates whether this is the default lead status value (True) or not (False) in the picklist.
MasterLabel	string	Master label for this lead status value. This display value is the internal label that does not get translated.
SortOrder	int	Number used to sort this value in the lead status picklist. These numbers are not guaranteed to be sequential, as some previous lead status values might have been deleted.

Usage

The LeadStatus object represents a value in the lead status picklist (see Lead Status Picklist on page 142). The lead status picklist provides additional information about the status of a Lead, such as whether a given status value represents a converted Lead. Your client application can invoke the query call on the LeadStatus object to retrieve the set of values in the lead status picklist, and then use that information while processing Lead objects to determine more information about a given lead. For example, the application could test whether a given lead is converted based on its `Status` value and the value of the `IsConverted` property in the associated LeadStatus object.

The LeadStatus object is read-only via the sforce API. With sufficient permissions, your client application can invoke the query and describeSObject calls on LeadStatus objects.

See Also

Concepts on page 101

MailMergeTemplate

Represents a mail merge template (a Microsoft Word document) used for performing mail merges for your organization.

Supported API Calls

create, update, delete, query, getDeleted, getUpdated, retrieve, describeSObject

Fields

For a complete list of fields in this object, see the Enterprise WSDL file for your organization and the salesforce.com online help.

Table 34: MailMergeTemplate Fields

Field	Data Type	Description
`Name`	string	Name of this mail merge template.
`Description`	string	Text description of this mail merge template. Up to 255 characters.
`Filename`	string	Filename of the Microsoft Word document that was uploaded as a mail merge template. Up to 255 characters in length.
`BodyLength`	int	Length of the Microsoft Word document.
`Body`	binary	Microsoft Word document to use as a mail merge template. Up to 5MB. Due to limitations with Microsoft Word mail merge templates, your client application can specify the Body field in the create call but not in the update call.
`LastUsedDate`	dateTime	Date and time when this MailMergeTemplate was last used.

Usage

Use the MailMergeTemplate object to manage mail merge templates for your organization. All users can view a MailMergeTemplate, but you need "Customize salesforce.com" permissions to modify it.

See Also

Concepts on page 101

Note

Represents a note, which is text associated with an Attachment, Contact, Contract, Opportunity, or custom object.

Supported API Calls

create, update, delete, query, search, getDeleted, getUpdated, retrieve, describeSObject

Fields

For a complete list of fields in this object, see the Enterprise WSDL file for your organization and the salesforce.com online help.

IsPrivate flag

The **IsPrivate** flag on a note is a field that influences the access rights for that note. If the note is marked private, only the note owner or a User with the "Modify All Data" permission can view the note or query it via the sforce API. Because of this, you can create an unusual situation for a regular user that does not have the "Modify All Data" permission. If a regular user sets the **IsPrivate** flag to True on a note that they do not own, then they can no longer query, update, or delete that note.

Usage

Use the Note object to manage notes for an Attachment, Contact, Contract, Opportunity, or custom object.

See Also

Concepts on page 101

Opportunity

Represents an opportunity, which is a sale or pending deal.

Supported API Calls

create, update, delete, query, search, getDeleted, getUpdated, retrieve, describeSObject

Fields

For a complete list of fields in this object, see the Enterprise WSDL file for your organization and the salesforce.com online help.

StageName Field

The **StageName** field controls several other fields on an Opportunity. Each of the fields can be directly set or implied by changing the **StageName** field. In addition, the **StageName** field is a picklist, so it has additional members in the describeSObject response to indicate how it affects the other fields. To obtain the stage name values in the picklist, a client application can invoke the query call on the OpportunityStage object. If the **StageName** is updated, then the **ForecastCategory** is automatically updated based on the stage-category mapping.

ForecastCategory

ForecastCategory is a restricted picklist field. It is implied, but not directly controlled, by the **StageName** field. You can override this field to a different value than is implied by the **StageName**.

The values of the **ForecastCategory** field are fixed enumerated values. The field labels are localized to the language of the user performing the operation, if localized versions of those labels are available for that language in the salesforce.com user interface.

IsClosed and IsWon Flags

The **IsClosed** and **IsWon** flags are directly controlled by the **StageName**. You can query and filter on these fields, but you cannot directly set them in a create or update request. Instead, you must set the **StageName** to a value that has the appropriate **IsClosed** and **IsWon** flags.

Probability Field

The Opportunity **Probability** field is implied, but not directly controlled, by the **StageName** field. You can override this field to a different value than what is implied by the **StageName**.

ExpectedRevenue Field

The **ExpectedRevenue** field is a read-only field that is equal to the product of the Opportunity **Amount** field and the **Probability**. You cannot directly set the **ExpectedRevenue** field, but you can indirectly set it by setting the **Amount** or **Probability** fields.

Amount Field

The Opportunity **Amount** field is normally a regular field, but it becomes implicitly read-only if the Opportunity has any line items. Any attempt to update the **Amount** of an Opportunity that has line items will be ignored. The update call will not be rejected, and other fields will be updated as specified, but the **Amount** will be unchanged.

CampaignId Field

The Opportunity **CampaignId** field is a cross-reference field that points to a Campaign object. The **CampaignId** field is defined only for those organizations that have Campaigns enabled as a feature. The User must have read access rights to the cross-referenced Campaign object in order to create or update that campaign into the **CampaignId** field on the Opportunity.

HasOpportunityLineItem Field

The Opportunity **HasOpportunityLineItem** field is a read-only field that indicates whether the Opportunity has associated line items. A value of True means that Opportunity line items have been created for the Opportunity.

Pricebook2Id and PricebookId Fields

The Opportunity **Pricebook2Id** field is a cross-reference field that points to a Pricebook2 object. The **Pricebook2Id** field indicates which Pricebook2 applies to this specific Opportunity. The **Pricebook2Id** field is defined only for those organizations that have Products enabled as a feature.

The **PricebookId** field, like the Pricebook [Deprecated] object, has been deprecated as of version 3.0 and is provided for backward compatibility only. Unless you need to continue referring to a Pricebook [Deprecated] object for this Opportunity in an existing client application, use the **Pricebook2Id** field instead, specifying the ID of the Pricebook2 object.

Note

You can specify values for only *one* field (**Pricebook2Id** or **PricebookId**)—not both fields. For this reason, both fields are declared nillable.

An Opportunity can only have Opportunity line items if the Opportunity has a price book. The opportunity line items must correspond to PricebookEntrys that are listed in the Opportunity's Pricebook2. However, you can insert Opportunity line items on an Opportunity that does not have an associated Pricebook2. For the first Opportunity line item that you insert on an Opportunity without a Pricebook2, the API automatically sets the **Pricebook2Id** field, if the Opportunity line item corresponds to a PricebookEntry in an active Pricebook2 that has a **CurrencyISOCode** field that matches the **CurrencyISOCode** field of the Opportunity. If the Pricebook2 is not active or the **CurrencyISOCode** fields do not match, then the sforce Web service returns an error.

You cannot update the **Pricebook2Id** or **PricebookId** fields if opportunity line items exist on the Opportunity. You must delete the line items before attempting to update the **PricebookId** field.

Currency Field

The `CurrencyISOCode` field exists only for multi-currency organizations. If the organization does not have the multi-currency feature enabled, the `CurrencyISOCode` field is not accessible.

If the organization is multi-currency and a Pricebook2 is specified on the Opportunity (i.e., the `PricebookId` field is not blank), then the currency value of the `CurrencyISOCode` field must match the currency of the PricebookEntry objects that are associated with any opportunity line items it has.

Usage

Use the Opportunity object to manage information about a sale or pending deal. To update an Opportunity, your client application needs "Edit" permission on Opportunities. Client applications can create, update, delete, and query Attachments associated with an opportunity via the sforce API. For a visual diagram of the relationships between Opportunity and other sforce objects, see Product and Schedule Objects on page 184.

Client applications can also create or update Opportunity objects by converting a Lead via the convertLead call..

See Also

OpportunityCompetitor on page 149
OpportunityHistory on page 150
OpportunityLineItem on page 151
OpportunityLineItemSchedule on page 153
Concepts on page 101

OpportunityCompetitor

Represents a competitor on an Opportunity.

Supported API Calls

create, update, delete, query, getDeleted, getUpdated, retrieve, describeSObject

Fields

For a complete list of fields in this object, see the Enterprise WSDL file for your organization and the salesforce.com online help.

Table 35: OpportunityCompetitor Fields

Field	Data Type	Description
`OpportunityID`	ID	ID of the associated Opportunity.
`CompetitorName`	string	Name of the competitor.
`Strengths`	string	Description of the competitor's strengths.
`Weaknesses`	string	Description of the competitor's weaknesses.

Usage

Use the OpportunityCompetitor object to manage competitors on an Opportunity, associating multiple competitors on a opportunity and specifying the strengths and weaknesses of each competitor.

See Also

Opportunity on page 147
Concepts on page 101

OpportunityContactRole

Represents the association between an Opportunity and a Contact, with a specified UserRole name applied to the contact.

Supported API Calls

create, update, delete, query, getDeleted, getUpdated, retrieve, describeSObject

Fields

For a complete list of fields in this object, see the Enterprise WSDL file for your organization and the salesforce.com online help.

OpportunityId Field

The `OpportunityId` field of an OpportunityContactRole is non-nullable, and it cannot be updated. You must provide a value for the `OpportunityId` on create. You cannot change it after it has been inserted.

ContactId Field

The sforce API applies user access rights to the associated Opportunity for an OpportunityContactRole, but not to the associated Contact. As such, the API may return rows from an OpportunityContactRole query that include `ContactId` values for contacts that the user does not have sufficient rights to access. It may also return `ContactId` values for contacts that have been deleted. In either case, the client must perform a query on the contact table for that `ContactId` value to determine whether the contact is accessible to the user and has not been deleted.

Usage

OpportunityContactRoles appear in the salesforce.com user interface on the Opportunity detail page. Like most other sforce objects, OpportunityContactRole records have their own unique ID that you use when updating or deleting an OpportunityContactRole.

You can create multiple relationships between the same Opportunity and a Contact. This action is not recommended, but the application does not prohibit it.

See Also

Concepts on page 101

OpportunityHistory

Represents the history of an Opportunity.

Supported API Calls

query, retrieve, describeSObject

Fields

For a complete list of fields in this object, see the Enterprise WSDL file for your organization and the salesforce.com online help.

Table 36: OpportunityHistory Fields

Field	Data Type	Description
`OpportunityID`	ID	ID of the associated Opportunity.
`Amount`	double	Estimated total sale amount. See Amount Field on page 148 for more information.
`CloseDate`	date	Date when the opportunity is expected to close.
`ExpectedRevenue`	double	Calculated revenue based on the `Amount` and `Probability` fields. See ExpectedRevenue Field on page 148 for more information.
`ForecastCategory`	string	Category that determines the column in which an opportunity is totaled in a forecast. See ForecastCategory on page 147 for more information.
`Probability`	double	Percentage of estimated confidence in closing the opportunity. See Probability Field on page 148 for more information.
`StageName`	string	Name of the current stage of the opportunity (for example, Prospect or Proposal). See StageName Field on page 147 for more information.

Usage

The OpportunityHistory object represents the history of a change to any of the major fields (listed above) of an Opportunity. To obtain information about how fast a particular opportunity is progressing, query all of the OpportunityHistory objects associated with a given Opportunity.

The OpportunityHistory object is read-only. The system generates a new OpportunityHistory object whenever a user or client application changes the value of any of the above fields; the then-current values of all of these major fields are saved in the newly-generated object.

Note that the OpportunityHistory object is automatically deleted if its parent Opportunity is deleted.

See Also

Opportunity on page 147
Concepts on page 101

OpportunityLineItem

Represents an opportunity line item, which is a member of the list of Product2s associated with an Opportunity, along with other information about those products on that opportunity.

Supported API Calls

create, update, delete, query, getDeleted, getUpdated, retrieve, describeSObject

Fields

For a complete list of fields in this object, see the Enterprise WSDL file for your organization and the salesforce.com online help.

Total Price and Unit Price Fields

The **TotalPrice** field specified in a create or update call is the total price of the OpportunityLineItem. The **UnitPrice** field represents the unit price for this OpportunityLineItem. In an update call, you can change one value or the other but not both in the same update call. For this reason, both are nillable, but you cannot set both to `null` in the same update call. The **TotalPrice** field is deprecated but remains to provide backward compatibility.

The salesforce.com user interface calculates the unit price as the total price of the OpportunityLineItem divided by the quantity listed for that line item. To insert the **TotalPrice** for an OpportunityLineItem via the API (given only a unit price and the quantity), calculate the **TotalPrice** field as the unit price multiplied by the quantity. The **TotalPrice** field is read-only if the OpportunityLineItem has a revenue schedule. If the OpportunityLineItem does not have a schedule or only has quantity schedule, the **TotalPrice** field can be updated.

ListPrice Field

The **ListPrice** field corresponds to the **UnitPrice** on the PricebookEntry that is associated with this line item, which can be in the standard pricebook or a custom pricebook. A client application can use this information to show whether the unit price (or sales price) of the line item differs from the pricebook entry list price.

Quantity Field

The **Quantity** field on the OpportunityLineItem object is read-only if the OpportunityLineItem has a quantity schedule, a revenue schedule, or both a quantity and a revenue schedule.

Has Revenue Schedule and Has Quantity Schedule Fields

The **HasRevenueSchedule** field on the OpportunityLineItem object is read-only, and is True if a revenue schedule has been created for the OpportunityLineItem.

The **HasQuantitySchedule** field on the OpportunityLineItem object is read-only, and is True if a quantity schedule has been created for the OpportunityLineItem.

If the OpportunityLineItem has a revenue schedule, the **Quantity** and **TotalPrice** fields cannot be updated. In addition, the **Quantity** field cannot be updated if the OpportunityLineItem has a quantity schedule. The sforce API ignores any attempt to update these fields. The update call will not be rejected; the updated values will simply be ignored.

PricebookEntryId and ProductId Fields

The OpportunityLineItem **PricebookEntryId** field is a cross-reference field that points to the ID of the PricebookEntry object associated with this OpportunityLineItem. The `PricebookEntryId` field is defined only for those organizations that have Products enabled as a feature.

The **ProductId** field, like the Product [Deprecated] object, has been deprecated as of version 3.0 and is provided for backward compatibility only. Unless you need to continue referring to a Product [Deprecated] object for this OpportunityLineItem in an existing client application, use the **PricebookEntryId** field instead, specifying the ID of the PricebookEntry object.

Note You can specify values for only *one* field (**PricebookEntryId** or **ProductId**)—not both fields. For this reason, both fields are declared nillable.

Usage

The Opportunity can only have OpportunityLineItems if the opportunity has a Pricebook2. An OpportunityLineItem must correspond to a Product2 that is listed in the opportunity's

Pricebook2. For information about inserting OpportunityLineItems for an Opportunity that does not have an associated Pricebook2 or any existing line items, see Effects on Opportunities on page 153.

The OpportunityLineItem object is defined only for those organizations that have Products enabled as a feature. If the organization does not have the Products feature, the OpportunityLineItem object does not appear in the describeGlobal call, and you cannot use describeSObject or query with the OpportunityLineItem object.

For a visual diagram of the relationships between OpportunityLineItem and other sforce objects, see Product and Schedule Objects on page 184.

Permissions

The user must have "Edit" permission on the Opportunity in order to create or update opportunity line items on that opportunity.

Effects on Opportunities

Opportunities that have associated OpportunityLineItems are affected in the following ways:

- Inserting an OpportunityLineItem increments the Opportunity **`Amount`** value by the **`TotalPrice`** of the OpportunityLineItem. Additionally, inserting an OpportunityLineItem increments the **`Expected Amount`** on the opportunity by the **`TotalPrice`** times the opportunity **`Probability`**.
- The opportunity **`Amount`** becomes a read-only field when the opportunity has line items. The API ignores any attempt to update this field on an opportunity with line items. The update call will not be rejected; the updated value will simply be ignored.
- You cannot update the **`PricebookId`** field or the **`CurrencyISOCode`** field on the opportunity if line items exist. The API rejects any attempt to update these fields on an opportunity with line items.
- When you create or update an OpportunityLineItem, the API verifies that the line item corresponds to a PricebookEntry in the Pricebook2 that is associated with the opportunity. If the opportunity does not have an associated Pricebook2, the sforce Web service automatically sets the pricebook on the opportunity if the line item corresponds to a PricebookEntry in an active Pricebook2, and if the PricebookEntry has a **`CurrencyISOCode`** field that matches the **`CurrencyISOCode`** field of the opportunity. If the Pricebook2 is not active or the **`CurrencyISOCode`** fields do not match, the sforce Web service returns an error.
- The opportunity **`HasLineItem`** field is set to True when an OpportunityLineItem is inserted for that opportunity.

See Also

OpportunityLineItemSchedule on page 153
Concepts on page 101

OpportunityLineItemSchedule

Represents information about the quantity, revenue distribution, and delivery dates for a particular OpportunityLineItem.

Supported API Calls

create, update, delete, query, getDeleted, getUpdated, retrieve, describeSObject

Fields

For a complete list of fields in this object, see the Enterprise WSDL file for your organization and the salesforce.com online help.

Type Field

The **Type** field specifies the type of the schedule.You must specify a **Type** value when inserting an OpportunityLineItemSchedule. Valid values are "Quantity," "Revenue," or "Both." However, the allowable values for a particular OpportunityLineItemSchedule depend on the product-level schedule preferences and whether the line item has any existing schedules. The following criteria must be met:

- The Product2 on which the OpportunityLineItem is based must have the appropriate **RevenueScheduleEnabled** and/or **QuantityScheduleEnabled** flags set to True.
- When you create a schedule for a line item that does not have any existing schedules, you can specify any valid value.
- If you create a schedule for a line item that already has existing schedules, the new schedule must be consistent with the existing schedules. The following matrix outlines the allowable values:

Table 37: Allowable type Values for Opportunity Line Item Schedules

Value of **HasRevenueSchedule** on line item	Value of **HasQuantitySchedule** on line item	Allowable **Type** Values
false	false	Revenue, Quantity, Both
false	true	Quantity
true	false	Revenue
true	true	Both

Quantity and Revenue Fields

The **Quantity** field specifies the total number of units to be scheduled in a quantity schedule. The **Revenue** field specifies the total price to be scheduled in a revenue schedule.

The allowable **Quantity** and **Revenue** field values depend on the value of the **Type** field:

Table 38: Allowable Quantity and Revenue Values for Line Item Schedules

Type Value	Allowable **Quantity** Value	Allowable **Revenue** Value
Revenue	Null	Non-null
Quantity	Non-null	Null
Both	Non-null	Non-null

The **Quantity** and **Revenue** fields have the following restrictions in the update call.

- For a schedule of **Type** "Quantity," you cannot update a null **Revenue** value to non-null. Likewise for a schedule of **Type** "Revenue," you cannot update a null **Quantity** value to non-null.
- You cannot null out the **Quantity** field for a schedule of **Type** "Quantity." Likewise you cannot null out the **Revenue** field for a schedule of **Type** "Revenue."
- You cannot null out either the **Revenue** or **Quantity** fields for a schedule of type "Both."

Usage

Sforce supports two types of OpportunityLineItemSchedules:

- quantity schedules

- revenue schedules

The user must have edit access rights on the Opportunity in order to create or update line item schedules on that opportunity.

Applies Only If Products and Annuities Features Are Enabled

The OpportunityLineItemSchedule object is defined only for those organizations that have the Products and Annuities features enabled. If the organization does not have the Products and Annuities features, the OpportunityLineItemSchedule object does not appear in the describeGlobal call, and you cannot use describeSObject or query with the OpportunityLineItemSchedule object.

Effects on Opportunities and Opportunity Line Items

OpportunityLineItemSchedules affect opportunities and opportunity line items in the following ways:

- Inserting an OpportunityLineItemSchedule of **`Type`** "Revenue" or "Quantity" increments the **`TotalPrice`** field on the OpportunityLineItem by the line item schedule **`Revenue`** amount. Inserting an OpportunityLineItemSchedule of **`Type`** "Quantity" or "Both" increments the **`Quantity`** field on the OpportunityLineItem by the line item schedule **`Quantity`** amount.
- The create call also affects the original opportunity as follows: 1) the Opportunity **`Amount`** is incremented the by OpportunityLineItemSchedule revenue amount; and 2) the Opportunity **`Expected Amount`** is incremented by the line item schedule amount multiplied by the Opportunity **`Probability`**.
- Deleting an OpportunityLineItemSchedule has a similar effect on the related OpportunityLineItem and Opportunity. Deleting a schedule decrements the OpportunityLineItem **`TotalPrice`** by the deleted line item schedule amount. The opportunity **`Amount`** is also decremented by the line item schedule amount, and the Opportunity **`Expected Amount`** is reduced by the line item schedule amount multiplied by the Opportunity **`Probability`**.

Deleting an Opportunity Line Item Schedule

Deleting the last remaining schedule will set the corresponding **`HasQuantitySchedule`** and/or **`HasRevenueSchedule`** flags to False on the parent line item.

See Also

OpportunityLineItem on page 151
Product2 *on page 166*
delete on page 52
Concepts on page 101

OpportunityShare

Represents a sharing entry on an Opportunity.

Supported API Calls

create, update, delete, query, getDeleted, getUpdated, retrieve, describeSObject

Fields

For a complete list of fields in this object, see the Enterprise WSDL file for your organization and the salesforce.com online help.

Table 39: OpportunityShare Fields

Field	Data Type	Description
`OpportunityId`	ID	ID of the Opportunity associated with this sharing entry. This field cannot be updated.
`UserOrGroupId`	ID	ID of the User or Group that has been given access to the Opportunity. This field cannot be updated.
`OpportunityAccessLevel`	string	Level of access that the User or Group has to the Opportunity. One of the following values: • **None** - User or Group cannot access the Opportunity. • **Read** - User or Group can only view the Opportunity. • **Edit** - User or Group can view or edit the Opportunity. • **All** - User or Group can view, edit, delete, and share the Opportunity with other users. This value is not valid for create and update calls. This field must be set to an access level that is higher than the organization's default access level for opportunities.
`RowCause`	string	Reason that this sharing entry exists. Read-only. One of the following values: • **Owner** - The User is the owner of the Opportunity or is in a UserRole above the Opportunity owner in the role hierarchy. • **Manual** - The User or Group has access because a User with "All" access manually shared the Opportunity with them. • **Rule** - The User or Group has access via an Opportunity sharing rule. • **ImplicitChild** - The User or Group has access to the Opportunity on the Account associated with this Opportunity. • **Team** - The User has access to the Opportunity because she or he is on the sales team for the Opportunity. The OpportunityTeamMember object for this Opportunity sets the access level. See OpportunityTeamMember on page 158 for more information.

Usage

The OpportunityShare object allows you to determine which users and groups can view and/or edit opportunities owned by other users. For more information, see Sharing on page 21.

If you attempt to create an OpportunityShare that matches an existing OpportunityShare record, the create call updates any modified fields and returns the existing record.

See Also

Group on page 140
Concepts

OpportunityStage

Represents the stage of an Opportunity in the sales pipeline, such as New Lead, Negotiating, Pending, Closed, and so on.

Supported API Calls

query, retrieve, describeSObject

Fields

For a complete list of fields in this object, see the Enterprise WSDL file for your organization and the salesforce.com online help.

Table 40: OpportunityStage Fields

Field	Data Type	Description
`DefaultProbability`	double	The default percentage estimate of the confidence in closing a specific opportunity for this opportunity stage value.
`Description`	string	Description of this opportunity stage value.
`ForecastCategory`	boolean	The default forecast category for this opportunity stage value. The forecast category automatically determines how opportunities are tracked and totaled in a forecast.
`IsActive`	boolean	Indicates whether this opportunity stage value is active (True) or not (False). Inactive opportunity stage values are not available in the picklist and are retained for historical purposes only.
`IsClosed`	boolean	Indicates whether this opportunity stage value represents a closed opportunity (True) or not (False). Multiple opportunity stage values can represent a closed opportunity.
`IsWon`	boolean	Indicates whether this opportunity stage value represents a won opportunity (True) or not (False). Multiple opportunity stage values can represent a won opportunity.
`MasterLabel`	string	Master label for this opportunity stage value. This display value is the internal label that does not get translated.
`SortOrder`	int	Number used to sort this value in the opportunity stage picklist. These numbers are not guaranteed to be sequential, as some previous opportunity stage values might have been deleted.

Usage

The OpportunityStage object represents a value in the opportunity stage picklist (see StageName Field on page 147). The opportunity stage picklist provides additional information about the stage of a Opportunity, such as its probability (see Probability Field on page 148) or forecast category (see ForecastCategory on page 147). Your client application can invoke the

query call on the OpportunityStage object to retrieve the set of values in the opportunity stage picklist, and then use that information while processing Opportunity objects to determine more information about a given opportunity. For example, the application could test whether a given opportunity is won or not based on its `StageName` value and the value of the `IsWon` property in the associated OpportunityStage object.

The OpportunityStage object is read-only via the sforce API. With sufficient permissions, your client application can invoke the query and describeSObject calls on OpportunityStage objects.

See Also

Concepts on page 101

OpportunityTeamMember

Represents an individual User on the sales team of a particular Opportunity.

Supported API Calls

create, update, delete, query, getDeleted, getUpdated, retrieve, describeSObject

Fields

For a complete list of fields in this object, see the Enterprise WSDL file for your organization and the salesforce.com online help.

Table 41: OpportunityTeamMember Fields

Field	Data Type	Description
`OpportunityId`	ID	ID of the Opportunity associated with this sales team. This field cannot be updated.
`UserId`	ID	ID of the User who is a member of the Opportunity's sales team. This field cannot be updated.
`TeamMemberRole`	string	Role that the team member has on the Opportunity. The valid values are set by the organization's administrator in the Sales Team Roles picklist.

Usage

If you attempt to insert an OpportunityTeamMember that matches an existing OpportunityTeamMember record, the create call updates any modified fields and returns the existing record.

In the salesforce.com user interface, users can set up a sales team for the opportunities they own. The sales team includes other users that are working on the Opportunity with them. The OpportunityTeamMember object is available only in organizations that have enabled the team selling functionality.

See Also

UserTeamMember on page 177
Concepts on page 101

Partner

Represents the association between two particular Accounts or between a particular Opportunity and an Account.

Supported API Calls

create, delete, query, getDeleted, retrieve, describeSObject

Fields

For a complete list of fields in this object, see the Enterprise WSDL file for your organization and the salesforce.com online help.

Table 42: Partner Fields

Field	Data Type	Description
`OpportunityId`	ID	ID of the Opportunity in a partner relationship between an Account and an Opportunity. Specifying this field when calling create creates an Opportunity Partner. If you specify the `AccountFromId` field, you cannot specify this field as well.
`AccountFromId`	ID	ID of the main Account in a partner relationship between two accounts. Specifying this field when calling create creates an Account Partner. If you specify the `OpportunityId` field, you cannot specify this field as well.
`AccountToId`	ID	ID of the Partner Account related to either an Opportunity or an Account. You must specify this field when creating an Opportunity Partner or an Account Partner.
`Primary`	boolean	Valid for Opportunity Partners only. Indicates that the Account is the primary Partner for the Opportunity. Only one Account can be marked as "primary" for an Opportunity. If you set the `Primary` flag to '1' upon insert of a new Opportunity Partner, any other existing primary partners for that Opportunity will automatically have their `Primary` flag set to False.
`Role`	string	UserRole that the Account has towards the related Opportunity or Account (e.g., "Consultant" or "Distributor").

Usage

All of the Partner fields are accessible in the describeSObject and query calls; see Fields on page 159. You must have the "View All Data" permission to access Partners via the API.

Each Account in the relationship is assigned a Role (e.g., "Consultant" or "Distributor") designating that Account's Role towards the related Account or Opportunity. A relationship between two Accounts is referred to as an Account Partner, and a relationship between an Opportunity and an Account is referred to as an Opportunity Partner.

Using the create call, you can insert the **OpportunityId** or **AccountFromId**, **AccountToId**, **Primary**, and **Role** fields. You cannot update Partners via the sforce API.

Using the create call, you can insert the **OpportunityId** or **AccountFromId**, **AccountToId**, **Primary**, and **Role** fields. When creating a Partner object, you must specify either the **OpportunityId** field or the **AccountFromId** field. Specifying the **OpportunityId** field creates an Opportunity Partner, and the **AccountFromId** field creates an Account Partner. You must always specify a value for the **AccountToId** field.

When you create an Account Partner, i.e., a relationship between two Accounts, the sforce API automatically creates a reverse partner relationship between those two Accounts. For example, if you create an Account Partner with "Acme, Inc." as the **AccountFromId** and "Acme Consulting" as the **AccountToId**, the API automatically creates a reverse partner with "Acme Consulting" as the **AccountFromId** and "Acme, Inc." as the **AccountToId**. In the reverse partner, the value of the **Role** field is set to the designated reverse Role value associated with the value of the **Role** field in the original Account Partner. In the salesforce.com user interface, system administrators can set up the valid Role values and their corresponding reverse Role values.

If you set the **Primary** flag to '1' upon insert of a new Opportunity Partner, any other existing primary partners for that Opportunity will automatically have their **Primary** flag set to False.

See Also

Concepts on page 101

PartnerRole

Represents a role for an account Partner, such as consultant, supplier, and so on.

Supported API Calls

query, retrieve, describeSObject

Fields

For a complete list of fields in this object, see the Enterprise WSDL file for your organization and the salesforce.com online help.

Table 43: PartnerRole Fields

Field	Data Type	Description
MasterLabel	string	Master label for this partner role value. This display value is the internal label that does not get translated.
ReverseRole	string	Name of the reverse role that corresponds to this partner role. For example, if the role is "subcontractor", then the reverse role might be "general contractor". In the user interface, assigning a partner role to an account creates a reverse partner relationship so that both accounts list the other as a partner.
SortOrder	int	Number used to sort this value in the partner role picklist. These numbers are not guaranteed to be sequential, as some previous partner role values might have been deleted.

Usage

The PartnerRole object represents a value in the partner role picklist. The partner role picklist provides additional information about the role of a Partner, such as their corresponding reverse role. Your client application can invoke the query call on the PartnerRole object to retrieve the set of values in the partner role picklist, and then use that information while processing PartnerRole objects to determine more information about a given partner role. For example, the application could determine the reverse role of a given partner role based on its `Role` value and the value of the `ReverseRole` property in the associated PartnerRole object.

The PartnerRole object is read-only via the sforce API. With sufficient permissions, your client application can invoke the query and describeSObject calls on PartnerRole objects.

See Also

Concepts on page 101

Pricebook [Deprecated]

Represents a price book that contains the list of Product [Deprecated]s that your organization sells.

Note The Pricebook object has been deprecated as of version 3.0 and is provided for backwards compatibility only. Unless you need to continue using the Pricebook object in an existing application, use Pricebook2 instead.

Supported API Calls

create, update, query, getUpdated, retrieve, describeSObject

Fields

For a complete list of fields in this object, see the Enterprise WSDL file for your organization and the salesforce.com online help.

Usage

Use the Pricebook object to query information about Pricebooks that have been configured for an organization. The purpose of the Pricebook object is to allow you to obtain valid Pricebook object IDs for use when querying or modifying Product [Deprecated]s through the API. See Product [Deprecated] on page 165 for more information.

The Pricebook object is defined only for those organizations that have Products enabled as a feature. If the organization does not have the Products feature, the Pricebook object does not appear in the describeGlobal call, and you cannot use describeSObject or query with the Pricebook object.

Products are the main constituents of Pricebooks. "Setting up a price book" via the sforce API usually means loading products into those Pricebooks. The usual way to configure Pricebooks via the API is:

1. Manually create the Pricebooks using the salesforce.com user interface.
2. Query the Pricebook object to obtain the Pricebook IDs.
3. To load products into those Pricebooks, call create or update using the sforce API.

See Also

Concepts on page 101

Pricebook2

Represents a price book that contains the list of products that your organization sells.

Note In this release, price books are represented by Pricebook2 objects. The Pricebook [Deprecated] object has been deprecated.

Supported API Calls

create, update, delete, query, getDeleted, getUpdated, retrieve, describeSObject

Fields

For a complete list of fields in this object, see the Enterprise WSDL file for your organization and the salesforce.com online help.

Table 44: Pricebook2 Fields

Field	Data Type	Description
Description	string	Describes this Pricebook2 object.
IsActive	boolean	Indicates whether this Pricebook2 object is active (True) or not (False). Although client applications can never delete Pricebook2 objects, they can set the object's `IsActive` flag to False. Inactive Pricebook2 objects are hidden in many areas in the salesforce.com user interface. You can change the `IsActive` flag on a Pricebook2 object as often as necessary.
IsStandard	boolean	Indicates whether this Pricebook2 object is the standard price book for the organization (True) or not (False). Every organization has one standard price book—all other price books are custom price books.
Name	string	Name of this Pricebook2 object. This field is read-only for the standard pricebook.

Usage

A price book is a list of products that your organization sells. Each organization has one standard price book that defines the standard or generic list price for each product or service that your organization sells. In addition, an organization can have multiple custom price books that can be used for specialized purposes, such as a discount price book, price books for different channels or markets, price books for select accounts or opportunities, and so on. While your client application can create, update, and delete custom price books, your client application can only update the standard price book. For some organizations, the standard price book might be the only price needed, but if you need to set up further price books, you can reference the standard price book when setting up list prices in custom price books.

Use the Pricebook2 object to query information about standard and custom price books that have been configured for your organization. A common use of the Pricebook2 object is to allow your client application to obtain valid Pricebook2 object IDs for use when configuring PricebookEntry objects via the sforce API. Your client application can query, create (custom, not standard), update, and delete Pricebook2 objects.

How PriceBook2, Product2, and PricebookEntry Objects Are Related

In the sforce API:

- Price books are represented by Pricebook2 objects (the Pricebook [Deprecated] object has been deprecated).
- Products are represented by Product2 objects (the Product [Deprecated] object has been deprecated).
- Each price book contains zero or more entries (represented by PricebookEntry objects) that specify the products that are associated with the price book. A price book entry defines the price for which you sell a product at a particular currency.

These objects are defined only for those organizations that have Products enabled as a feature. If the organization does not have the Products feature enabled, the Pricebook2 object does not appear in the describeGlobal call, and you cannot access it via the sforce API.

For a visual diagram of the relationships between Pricebook2 and other sforce objects, see Product and Schedule Objects on page 184.

Setting Up a Price Book

The process of setting up a price book via the sforce API usually means:

1. Initially loading product data from your organization into Product2 objects (calling create for each product that you want to add).
2. For each Product2 object, creating a PricebookEntry that links the Product2 object to the standard Pricebook2. You need to define a standard price for a product at a given currency (if you have multi-currency enabled), before defining a price for that product in the same currency in a custom pricebook.
3. Creating a custom Pricebook2.
4. Querying the Pricebook2 object to obtain the Pricebook2 IDs.
5. For each Pricebook2 object, creating a PricebookEntry for every Product2 that you want to add, specifying unique properties for each PricebookEntry (such as the `UnitPrice` and `CurrencyIsoCode`) as needed.

See Also

Concepts on page 101
Product and Schedule Objects on page 184

PricebookEntry

Represents a product entry (an association between a Pricebook2 and Product2) in a pricebook.

Supported API Calls

create, update, delete, query, getDeleted, getUpdated, retrieve, describeSObject

Fields

For a complete list of fields in this object, see the Enterprise WSDL file for your organization and the salesforce.com online help.

Table 45: PricebookEntry Fields

Field	Data Type	Description
`CurrencyIsoCode`	string	Available only for organizations with the multi-currency feature enabled. Contains the ISO code for any currency allowed by the organization.
`IsActive`	boolean	Indicates whether this PricebookEntry object is active (True) or not (False). Although you can never delete PricebookEntry objects, your client application can set the object's `IsActive` flag to False. Inactive PricebookEntry objects are hidden in many areas in the salesforce.com user interface. You can change the `IsActive` flag on a PricebookEntry object as often as necessary.
`Name`	string	Name of this PricebookEntry object. This read-only field references the value in the Name field of the Product2 object.
`Pricebook2Id`	ID	ID of the Pricebook2 object with which this PricebookEntry is associated. Required field. This field must be specified in the create call. It cannot be changed in an update call.
`Product2Id`	ID	ID of the Product2 object with which this PricebookEntry is associated. Required field. This field must be specified in the create call. It cannot be changed in an update call.
`ProductCode`	string	Product code for this PricebookEntry object. This read-only field references the value in the `ProductCode` field of the associated Product2 object.
`UnitPrice`	double	Unit price for this PricebookEntry object. You can specify a `UnitPrice` only if `UseStandardPrice` is set to False.
`UseStandardPrice`	boolean	Indicates whether this PricebookEntry object uses the standard price defined in the standard Pricebook2 object (True) or not (False). If set to True, then this field is read-only and it references the `UnitPrice` value in the associated Product2 object. For PricebookEntry objects associated with the standard Pricebook2 object, `UseStandardPrice` *must* be set to True.

Usage

Use the PricebookEntry object to define the association between your organization's products (Product2 object) and your organization's standard price book or to other, custom-defined price books (Pricebook2 objects). Using PricebookEntry objects allows you to configure the standard price for a product in this price book or to override it with a custom value and, for multi-currency organizations, to specify different currencies as well.

When your client application calls create, it must specify the IDs of the associated Pricebook2 object and Product2 object. Once created, your client application cannot update these IDs.

The PricebookEntry object is defined only for those organizations that have Products enabled as a feature. If the organization does not have the Products feature enabled, then the PricebookEntry object does not appear in the describeGlobal call, and you cannot access it via the sforce API.

For a visual diagram of the relationships between PricebookEntry and other sforce objects, see Product and Schedule Objects on page 184 and How PriceBook2, Product2, and PricebookEntry Objects Are Related on page 163.

See Also

Concepts on page 101

Product [Deprecated]

Represents a product that your organization sells. A product is member of the list of items in a Pricebook [Deprecated].

Note The Product object has been deprecated as of version 3.0 and is provided for backwards compatibility only. Unless you need to continue using the Product object in an existing application, use Product2 instead.

Supported API Calls

create, update, query, getUpdated, retrieve, describeSObject

Fields

For a complete list of fields in this object, see the Enterprise WSDL file for your organization and the salesforce.com online help.

PriceBookId Field

The **`PricebookId`** field indicates the Pricebook [Deprecated] on which the Product is configured. When inserting Products, the **`PricebookId`** field must be set, and its value must be valid and non-blank. You cannot update the **`PricebookId`** field during an update call. You can query the Pricebook [Deprecated] object to obtain valid Pricebook [Deprecated] object IDs for your organization after they have been created using the salesforce.com user interface.

IsActive Flag

The **`IsActive`** flag indicates whether a Product is active. Although you can never delete Products, you can set a Product's **`IsActive`** flag to False. Inactive Products are hidden in many areas in the salesforce.com user interface. You can change a Product's **`IsActive`** flag as often as necessary.

Schedule Fields

The Product object has several fields that are only used for schedules (a.k.a, annuities). Sforce supports two types of schedules on Products - quantity schedules and revenue schedules. Schedules are available only for those organizations that have the Products and Annuities features enabled. If the organization does not have the Annuities feature, the schedule fields do not appear in the DescribeSObjectResult, and you cannot query, create, or update the fields.

Schedule Enabled Flags When enabling the Annuities feature, organizations can decide whether to enable quantity schedules, revenue schedules, or both. In addition, you can use the sforce API to control quantity and revenue scheduling at the product level via the **`RevenueScheduleEnabled`** and **`QuantityScheduleEnabled`** flags. A value of True for either flag indicates that the Product and any OpportunityLineItems can have a schedule of that type. These flags can be set via a create or update call.

Default Schedule Fields The remaining Product schedule fields define default schedules for the object. The sforce Web service uses the default schedule values to create an OpportunityLineItemSchedule when an OpportunityLineItem is created for the Product. The sforce API ignores default schedules on a product when a line item referencing it is created.

The following table lists the default schedule fields and their valid values.

Table 46: Default Schedule Fields on Products

Field	Valid Values
`RevenueScheduleType`	None, Divide, Repeat
`RevenueInstallmentPeriod`	None, Daily, Weekly, Monthly, Quarterly, Yearly
`NumberOfRevenueInstallments`	An integer from 1 to 100
`QuantityScheduleType`	None, Divide, Repeat
`QuantityInstallmentPeriod`	None, Daily, Weekly, Monthly, Quarterly, Yearly
`NumberOfQuantityInstallments`	An integer from 1 to 100

When you attempt to set the schedule fields via a create or update call, the sforce API applies cross-field integrity checks. The integrity requirements are:

- If the schedule type is set to "None," the installment period and number of installments must be null.
- If the schedule type is set to any value other than "None," the installment period and number of installments must be non-null.

Inserts or updates that fail these integrity checks are rejected with an error.

The `RevenueScheduleType`, `RevenueInstallmentPeriod`, `QuantityScheduleType`, and `QuantityInstallmentPeriod` fields are restricted picklist fields and are available only if the organization has the Annuities feature enabled.

Usage

The Product object allows you to query, create, and update (but not delete) Products on Pricebooks. These operations constitute the main configuration necessary for Pricebooks (see Pricebook [Deprecated] on page 161). Products can be queried, inserted, and updated via the sforce API, but they cannot be deleted through the API or any other means. See IsActive Flag on page 165 for more information. Because Products can never be deleted, please exercise caution when creating them.

The Product object is defined only for those organizations that have Products enabled as a feature. If the organization does not have the Products feature, the Product object does not appear in the describeGlobal call, and you cannot use describeSObject or query with the Product object.

See Also

OpportunityLineItem on page 151
OpportunityLineItemSchedule on page 153
Concepts on page 101

Product2

Represents a product that your organization sells.

Note In this release, products are represented by Product2 objects. The Product [Deprecated] object has been deprecated.

Supported API Calls

create, update, delete, query, getDeleted, getUpdated, retrieve, describeSObject

Fields

For a complete list of fields in this object, see the Enterprise WSDL file for your organization and the salesforce.com online help.

Table 47: Product2 Fields

Field	Data Type	Description
Description	string	Describes this Product2 object.
Family	string	Name of the product family associated with this Product2 object. Product families are configured as picklists in the salesforce.com user interface. To obtain a list of valid values, your client application can call describeSObject and process the DescribeSObjectResult for the values associated with the `Family` field.
IsActive	boolean	Indicates whether this Product2 object is active (True) or not (False). Although you can never delete Product2 objects, you can set the object's `IsActive` flag to False. Inactive Product2s are hidden in many areas in the salesforce.com user interface. You can change the `IsActive` flag on a Product2 object as often as necessary.
Name	string	Default name of this Product2 object.
ProductCode	string	Default product code for this Product2 object. The product code naming pattern is defined by your organization.

Schedule Fields

The Product2 object has several fields that are only used for schedules (a.k.a, annuities). The sforce API supports two types of schedules on Product2 objects:

- quantity schedules
- revenue schedules

Schedules are available only for those organizations that have the Products and Annuities features enabled. If the organization does not have the Annuities feature, the schedule fields do not appear in the DescribeSObjectResult, and you cannot query, create, or update the fields.

Schedule Enabled Flags

When enabling the Annuities feature, organizations can decide whether to enable quantity schedules, revenue schedules, or both. In addition, you can use the sforce API to control quantity and revenue scheduling at the product level via the **`CanUseQuantitySchedule`** and **`CanUseRevenueSchedule`** flags. A value of True for either flag indicates that the Product and any OpportunityLineItems can have a schedule of that type. These flags can be set via a create or update call.

Default Schedule Fields

The remaining Product2 schedule fields define default schedules for the object. Sforce uses the default schedule values to create an OpportunityLineItemSchedule when an OpportunityLineItem is created for the Product.

The following table lists the default schedule fields and their valid values (all fields are also nillable).

Table 48: Default Schedule Fields on Products

Field	Valid Values
`RevenueScheduleType`	Divide, Repeat.
`RevenueInstallmentPeriod`	Daily, Weekly, Monthly, Quarterly, Yearly
`NumberOfRevenueInstallments`	Integer between 1 to 150, inclusive.
`QuantityScheduleType`	Divide, Repeat
`QuantityInstallmentPeriod`	Daily, Weekly, Monthly, Quarterly, Yearly
`NumberOfQuantityInstallments`	Integer between 1 to 150, inclusive.

When you attempt to set the schedule fields via a create or update call, the sforce API applies cross-field integrity checks. The integrity requirements are:

- If the schedule type is nil, then the installment period and number of installments must be nil.
- If the schedule type is set to any value, then the installment period and number of installments must be non-nil.

Any create or update calls that fail these integrity checks are rejected with an error.

These default schedule fields, as well as `CanUseQuantitySchedule` and `CanUseRevenueSchedule`, are restricted picklist fields and are available only if the organization has the Annuities feature enabled:

Usage

Use Product2 objects to define the default product information for your organization. The Product2 object is associated by reference with Pricebook2 objects via PricebookEntry objects. The same product can be represented in different price books as price book entries. In fact, the same product can be represented multiple times (as separate PricebookEntry objects) in the same price book with different prices and/or currencies. A Product can only have one price for a given currency within the same Pricebook. To be used in custom price books, all standard prices must be added as price book entries to the standard price book.

Use the Product2 object to query information about the products that have been configured for your organization. A common use of the Product2 object is to allow your client application to obtain valid Product2 object IDs for use when configuring PricebookEntry objects via the sforce API. Your client application can query, create (custom, not standard), update, and delete Pricebook2 objects. You can also change the `IsActive` flag in create or update calls.

The Product2 object is defined only for those organizations that have Products enabled as a feature. If the organization does not have the Products feature, the Product2 object does not appear in the describeGlobal call, and you cannot use describeSObject or query with the Product object.

For a visual diagram of the relationships between Product2 and other sforce objects, see Product and Schedule Objects on page 184 and How PriceBook2, Product2, and PricebookEntry Objects Are Related on page 163.

See Also

Concepts on page 101

Profile

Represents a profile, which defines a set of permissions to perform different operations, such as querying, adding, updating, or deleting information.

Supported API Calls

create, update, delete, query, getDeleted, getUpdated, retrieve, describeSObject

Fields

For a complete list of fields in this object, see the Enterprise WSDL file for your organization and the salesforce.com online help.

Usage

Use the Profile object to query the set of currently configured Profiles in the organization. Your client application can use Profile objects to obtain valid Profile IDs for use when querying or modifying Users through the API. Your client application can query, create, update, and delete Profiles.

See Also

Concepts on page 101

RecordType

Represents a record type.

Supported API Calls

create, update, query, getUpdated, retrieve, describeSObject

Fields

For a complete list of fields in this object, see the Enterprise WSDL file for your organization and the salesforce.com online help.

Table 49: RecordType Fields

Field	Data Type	Description
`IsActive`	boolean	Indicates whether the RecordType is active or not. Only active RecordTypes can be used.
`Name`	string	Name of the RecordType.
`TableEnumOrIdName`	string	Object to which this RecordType applies. Valid values include "lead," "contact," "account," and other objects that support RecordTypes. A particular RecordType can apply to only one type of object.

Usage

Use the RecordType object to offer different BusinessProcesses and subsets of picklists values to different users based on their particular Profile. Your client application can invoke the describeSObject and query calls on RecordType objects. Record types are read-only in the sforce API.

The following objects have a **RecordTypeId** field: Account, Campaign, Case, Contact, Contract, Lead, Opportunity, and Solution. Client applications can set this field in create or update calls on these objects, specifying a valid record type ID associated with these objects. A client application can retrieve the list of valid record type IDs for a given object by calling query on the RecordType object. For more information, see RecordTypeId fields on page 109.

See Also

Concepts on page 101
Record Type Objects on page 184

Scontrol

Represents an sforce control, which is custom content that is hosted by the server but executed by client applications.

Supported API Calls

create, update, delete, query, getDeleted, getUpdated, retrieve, describeSObject

Fields

For a complete list of fields in this object, see the Enterprise WSDL file for your organization and the salesforce.com online help.

Table 50: Scontrol Fields

Field	Data Type	Description
Name	string	Name of this Scontrol.
Description	string	Description of this Scontrol.
HtmlWrapper	string	HTML page that will be delivered when the user views this Scontrol. This HTML page can be the entire content of the Scontrol, or it can reference the binary. Up to 1048576 characters.
Binary	binary	Binary content of this Scontrol, such as an ActiveX control or a Java archive. Up to 5MB. Can be specified when your client application calls create but not update.

Usage

Use Scontrol objects to manage custom content on the sforce Web service that is executed by client applications. All users can view Scontrol objects, but "Customize salesforce.com" permission is required to create or update Scontrol objects. Your organization must be using Enterprise Edition and be configured with sforce controls enabled.

See Also

Concepts on page 101

Solution

Represents a solution, which is a detailed description of a customer issue and the resolution of that issue.

Supported API Calls

create, update, delete, query, getDeleted, getUpdated, retrieve, describeSObject

Fields

For a complete list of fields in this object, see the Enterprise WSDL file for your organization and the salesforce.com online help.

Title Field

The `Title` field is required. If a client application creates a new Solution and a value for this field is unspecified, then the sforce Web service automatically specifies a hypen (-), the default value for this field.

SolutionNumber Field

The `SolutionNumber` field is assigned automatically when a Solution is created. It cannot be set directly, and it cannot be modified after the Solution is created.

IsPublished Flag

Solutions can have a special status called "published." (See the salesforce.com online help documentation for more information on publishing Solutions.) A Solution's published state does not affect how you can use a Solution, or whether you can query, update, or delete it.

IsPublishedPkb Flag

Indicates whether the solution has been published in the public knowledge base (True) or not (False).

Status Field and IsReviewed Flag

The `Status` field is a picklist field that also directly controls the `IsReviewed` flag. You cannot directly set the `IsReviewed` flag, but you can set it indirectly by setting the `Status` field. Each predefined `Status` field value implies an `IsReviewed` flag value. To obtain the solution status values in the picklist, a client application can invoke the query call on the SolutionStatus object.

Usage

Use Solution objects to manage your organization's solution knowledge base. Client applications can create, update, delete, and query Attachments associated with a solution via the sforce API.

See Also

Concepts on page 101

SolutionStatus

Represents the status of a Solution, such as Draft, Reviewed, and so on.

Supported API Calls

query, retrieve, describeSObject

Fields

For a complete list of fields in this object, see the Enterprise WSDL file for your organization and the salesforce.com online help.

Table 51: SolutionStatus Fields

Field	Data Type	Description
`IsDefault`	boolean	Indicates whether this is the default solution status value (True) or not (False) in the picklist. Only one value can be the default value.
`IsReviewed`	boolean	Indicates whether this solution status value represents a reviewed Solution (True) or not (False). Multiple solution status values can represent a reviewed Solution.
`MasterLabel`	string	Master label for this solution status value. This display value is the internal label that does not get translated.
`SortOrder`	int	Number used to sort this value in the solution status picklist. These numbers are not guaranteed to be sequential, as some previous solution status values might have been deleted.

Usage

The SolutionStatus object represents a value in the solution status picklist. The solution status picklist provides additional information about the status of a Solution, such as whether a given status value represents a reviewed or unreviewed solution. Your client application can invoke the query call on the SolutionStatus object to retrieve the set of values in the solution status picklist, and then use that information while processing Solution objects to determine more information about a given solution. For example, the application could test whether a given case has been reviewed or not based on its `Status` value and the value of the `IsReviewed` property in the associated SolutionStatus object.

The SolutionStatus object is read-only via the sforce API. With sufficient permissions, your client application can invoke the query and describeSObject calls on SolutionStatus objects. You cannot create, update, or delete a SolutionStatus object via the sforce API.

See Also

Concepts on page 101

Task

Represents a task.

Supported API Calls

create, update, delete, query, search, getDeleted, getUpdated, retrieve, describeSObject

Fields

For a complete list of fields in this object, see the Enterprise WSDL file for your organization and the salesforce.com online help.

WhoId and WhatId Fields

The task object has `WhoId` and `WhatId` fields that function like the same fields on the Event object. See Event on page 136 for more information.

Priority Field

The Priority field is a picklist that indicates the important of a task (such as high, normal, or low priority). To obtain the task priority values in the picklist, a client application can invoke the query call on the TaskStatus object.

Status Field and Closed Flag

The `Status` field is a picklist that directly controls the `Closed` flag. You cannot directly set the `Closed` flag, but you can set it indirectly by setting the `Status` field. Each predefined `Status` field value implies a `Closed` flag value. To obtain the task status values in the picklist, a client application can invoke the query call on the TaskStatus object.

ActivityDate Field

The due date information for the task object is contained in the `ActivityDate` field. This field is a date field with a timestamp that is always set to midnight in the GMT/UTC time zone. The timestamp is not relevant, and you should not attempt to alter it to account for any time zone differences. For more information, see Date Field Type on page 103.

Usage

Archived Task objects are not accessible via the sforce API. See Archived Activities on page 137.

See Also

Concepts on page 101

TaskPriority

Represents the priority (importance) of a Task, such as High, Normal, or Low.

Supported API Calls

query, retrieve, describeSObject

Fields

For a complete list of fields in this object, see the Enterprise WSDL file for your organization and the salesforce.com online help.

Table 52: TaskPriority Fields

Field	Data Type	Description
`IsDefault`	boolean	Indicates whether this is the default task priority value (True) or not (False) in the picklist. Only one value in the picklist can be the default value.
`IsHighPriority`	boolean	Indicates whether this task priority value represents a high priority Task (True) or not (False). Multiple task priority values can represent a high-priority Task.
`MasterLabel`	string	Master label for this task priority value. This display value is the internal label that does not get translated.
`SortOrder`	int	Number used to sort this value in the task priority picklist. These numbers are not guaranteed to be sequential, as some previous task priority values might have been deleted.

Usage

The TaskPriority object represents a value in the task priority picklist. The task priority picklist provides additional information about the importance of a Task, such as whether a given priority value represents a high priority. Your client application can invoke the query call on the TaskPriority object to retrieve the set of values in the task priority picklist, and then use that information while processing Task objects to determine more information about a given task. For example, the application could test whether a given task is high priority based on its `Priority` value and the value of the `IsHighPriority` property in the associated TaskPriority object.

The TaskPriority object is read-only via the sforce API. With sufficient permissions, your client application can invoke the query and describeSObject calls on TaskPriority objects.

See Also

Concepts on page 101

TaskStatus

Represents the status of a Task, such as Not started, Completed, or Closed.

Supported API Calls

query, retrieve, describeSObject

Fields

For a complete list of fields in this object, see the Enterprise WSDL file for your organization and the salesforce.com online help.

Table 53: TaskStatus Fields

Field	Data Type	Description
`IsClosed`	boolean	Indicates whether this task status value represents a closed Task (True) or not (False). Multiple task status values can represent a closed Task.
`IsDefault`	boolean	Indicates whether this is the default task status value (True) or not (False) in the picklist.
`MasterLabel`	string	Master label for this task status value. This display value is the internal label that does not get translated.
`SortOrder`	int	Number used to sort this value in the task status picklist. These numbers are not guaranteed to be sequential, as some previous task status values might have been deleted.

Usage

The TaskStatus object represents a value in the task status picklist. The task status picklist provides additional information about the status of a Task, such as whether a given status value represents an open or closed task. Your client application can invoke the query call on the TaskStatus object to retrieve the set of values in the task status picklist, and then use that information while processing Task objects to determine more information about a given task. For example, the application could test whether a given task is open or closed based on its `Status` value and the value of the `IsClosed` property in the associated TaskStatus object.

The TaskStatus object is read-only via the sforce API. With sufficient permissions, your client application can invoke the query and describeSObject calls on TaskStatus objects.

See Also

Concepts on page 101

User

Represents a user in your organization.

Supported API Calls

create, update, query, search, getUpdated, retrieve, describeSObject

Fields

For a complete list of fields in this object, see the Enterprise WSDL file for your organization and the salesforce.com online help.

Username Field

The **`Username`** field contains the name that a User enters to log into the sforce API or the salesforce.com user interface. The **`Username`** must be in the form of an email address. It must also be unique across all sforce instances. If you try to create or update a User with a duplicate **`Username`**, the operation is rejected and fault code 1229 "duplicate username" is returned.

Each inserted User also counts as a license in sforce. Every organization has a maximum number of licenses. If you attempt to exceed the maximum number of licenses by inserting Users, the create call is rejected and fault code 1230 "license limit exceeded" is returned.

IsActive Flag

The `IsActive` flag on the User object determines whether the User has access to log in. You can modify a User's active status using the salesforce.com user interface or via the sforce API.

Timezone Field

The `Timezone` field is a restricted picklist field. A User's time zone affects the offset used when displaying or entering times in the user interface. However, the sforce API does not use a User's time zone when querying or setting values.

The `Timezone` field values are named using region and key city, according to ISO standards. It can often be more convenient to manually set a User's time zone in the user interface, and then use that value for inserting or updating other Users via the API.

Locale Field

The `Locale` field is a restricted picklist field. The value of the `Locale` field affects formatting and parsing of values, especially numeric values, in the user interface. It does not affect the operation of the sforce API.

The `Locale` field values are named according to the language, and country if necessary, using two-letter ISO codes. The set of names is based on the ISO standard. It can often be more convenient to manually set a User's `Locale` in the user interface, and then use that value for inserting or updating other Users via the sforce API.

Usage

Use the User object to query information about users and to provision and modify users in your organization. Unlike with other objects, the records in the User table represent actual users—not data owned by users.

All Users have access to use query or describeSObject with User objects. To create or update a User object, you must log in with "Manage Users" permission.

Disabling Users

You cannot delete Users in the salesforce.com user interface or the sforce API. To disable a User, deactivate that User in the salesforce.com user interface. Because Users can never be deleted, we recommend that you exercise caution when creating them.

Passwords

For security reasons, you cannot query Users' passwords via the API or the salesforce.com user interface. However, the sforce API allows you to set and "reset" Users' passwords using the setPassword and resetPassword calls.

The password lockout status and the ability to reset a User's locked-out status is not available via the API. You must check and reset a User's password lockout status using the salesforce.com user interface.

See Also

getUserInfo on page 95
Concepts on page 101

UserRole

Represents a user role in your organization.

Note This object was called "Role" in previous versions of the sforce API documentation.

Supported API Calls

create, update, delete, query, search, getDeleted, getUpdated, retrieve, describeSObject

Fields

For a complete list of fields in this object, see the Enterprise WSDL file for your organization and the salesforce.com online help.

Usage

Use the UserRole object to query the set of currently configured user roles in your organization. Use it in your client application to obtain valid UserRole IDs to use when querying or modifying a User.

All Users have access to invoke query or describeSObject with the UserRole object. If your client application logs in with "Modify All Data" access, it can query, create, update, and delete Roles.

See Also

Concepts on page 101

UserTeamMember

Represents a single User on the default sales team of another user.

Supported API Calls

create, update, delete, query, search, getDeleted, getUpdated, retrieve, describeSObject

Fields

For a complete list of fields in this object, see the Enterprise WSDL file for your organization and the salesforce.com online help.

Table 54: UserTeamMember Fields

Field	Data Type	Description
`OwnerId`	ID	ID of the User who owns the default sales team. This field cannot be updated.
`UserId`	ID	ID of the User who is a member of the default sales team. This field cannot be updated.

Table 54: UserTeamMember Fields (Continued)

Field	Data Type	Description
`OpportunityAccessLevel`	string	Level of access that the team member has to opportunities for which the User has added his or her default sales team. One of the following values: • **None** - User cannot access the Opportunity. • **Read** - User can only view the Opportunity. • **Edit** - User can view or edit the Opportunity. • **All** - User can view, edit, delete, and share the Opportunity with other Users. This value is not valid for create and update calls. This field must be set to an access level that is higher than the organization's default access level for opportunities.
`TeamMemberRole`	string	Role that the team member has on opportunities for which the User has added his or her default sales team. The valid values are set by the organization's administrator in the Sales Team Roles picklist.

Usage

If you attempt to insert a UserTeamMember that matches an existing UserTeamMember record, the create call updates any modified fields and returns the existing record.

Users can set up their default sales team to include the other Users that typically work with them on opportunities. The UserTeamMember object is available only in organizations that have enabled the team selling functionality.

See Also

OpportunityTeamMember on page 158
Concepts on page 101

WebLink

Represents a web link to an URL or Scontrol.

Supported API Calls

create, update, delete, query, search, getDeleted, getUpdated, retrieve, describeSObject

Fields

For a complete list of fields in this object, see the Enterprise WSDL file for your organization and the salesforce.com online help.

Table 55: Web Link Fields

Field	Data Type	Description
Encoding Key	string	Encoding of parameters on the URL link.
HasMenuBar	boolean	Indicates whether the popup window shows a menubar (`true`) or not (`false`).
HasScrollbars	boolean	Indicates whether the popup window shows scroll bars (`true`) or not (`false`).
HasToolbar	boolean	Indicates whether the popup window shows browser toolbars (`true`) or not (`false`). Toolbars normally contain navigation buttons like Back, Forward, and Print.
Height	int	Height of the popup (in pixels).
IsResizable	boolean	Indicates whether users are allowed to resize the popup window (`true`) or not (`false`).
LinkType	string	Type of link (`Scontrol` or `URL`).
Name	string	Name to display on page.
OpenType	string	How the Web Link opens when clicked in a browser—`NewWindow`, `Sidebar`, or `NoSidebar`.
PageEnumOrId	string	Page on which to display the Web Link. One of the following values: • `AccountDetail` • `CampaignDetail` • `CaseDetail` • `ContactDetail` • `ContractDetail` • `EventDetail` • `LeadDetail` • `OpportunityDetail` • `Product2Detail` • `SolutionDetail` • `TaskDetail` • `UserDetail`
Position	string	Location on the screen where the popup should open—`TopLeft`, `FullScreen`, or `None`.

Table 55: Web Link Fields (Continued)

Field	Data Type	Description
`ScontrolId`	ID	ID of the sforce control (Scontrol) to link to. Can include salesforce.com fields as tokens within the sforce control.
`ShowsLocation`	boolean	Indicates whether the popup window shows the browser's address bar containing the URL (`true`) or not (`false`).
`ShowsStatus`	boolean	Show the status bar at the bottom of the browser.
`Url`	string	URL of the page to link to. Can include salesforce.com fields as tokens within the URL.
`Width`	int	Width of the popup (in pixels).

Usage

Use the WebLink object to programmatically manage *web links*, which allow client applications to integrate salesforce.com data with external URLs, an organization's intranet, or other back-end office systems. To create a WebLink, the client application must be logged in with "Customize salesforce.com" permission. A WebLink can point to:

- An external URL, such as www.google.com or your company's intranet.
- An sforce control in the sforce control library, such as a Java applet or Active-X control.

Web links can include salesforce.com fields as tokens within the URL or sforce control.

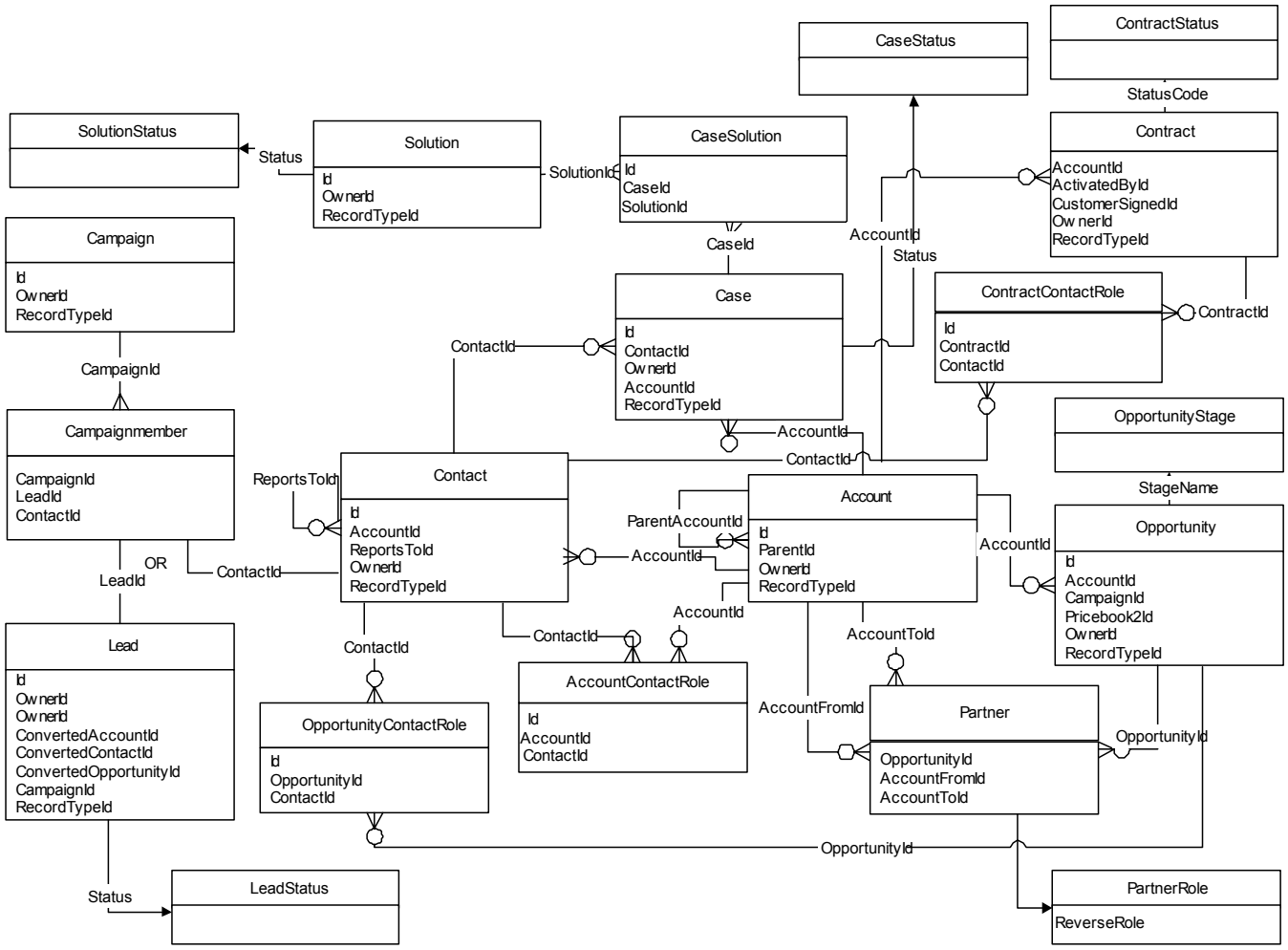
See Also

Concepts on page 101

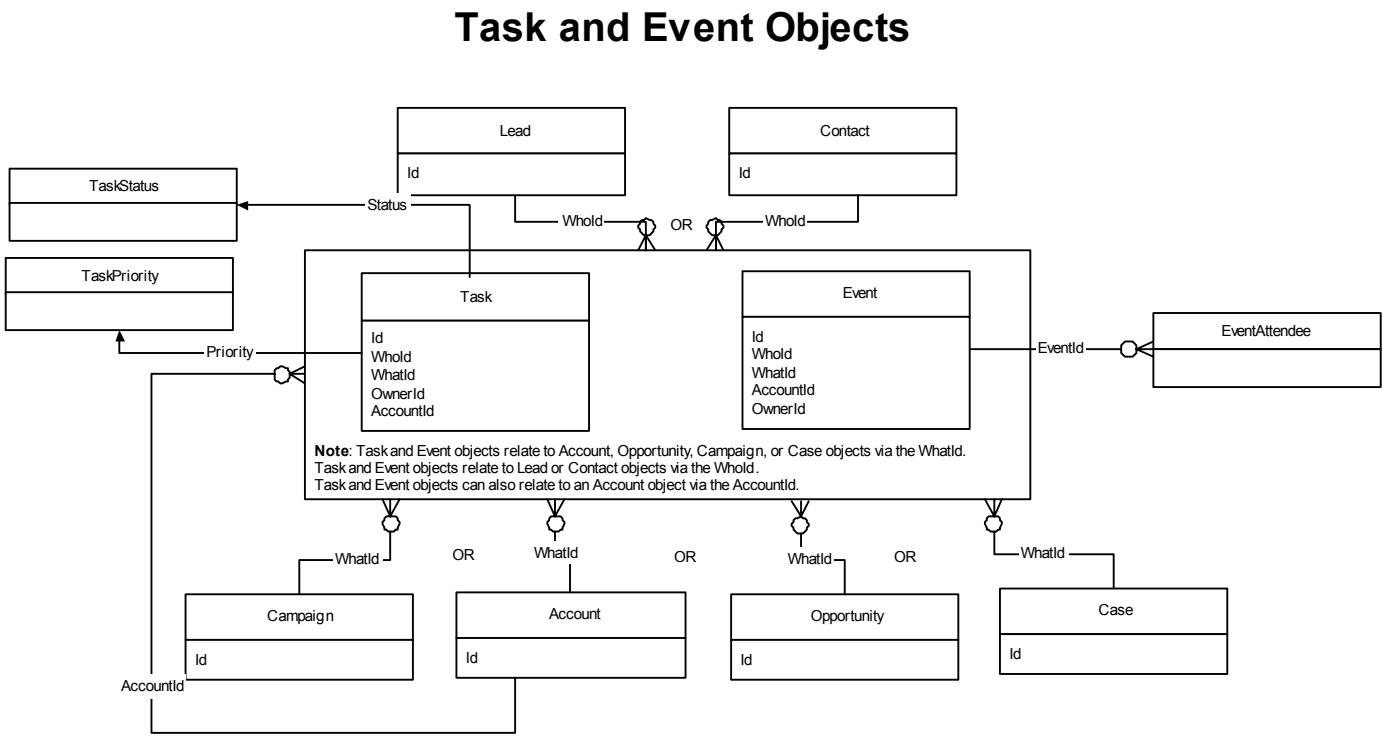
CHAPTER 6: Entity Relationship Diagrams

This topic describes the entity relationship diagrams (ERDs) for sforce objects in the sforce Web services API. It includes the following sections:

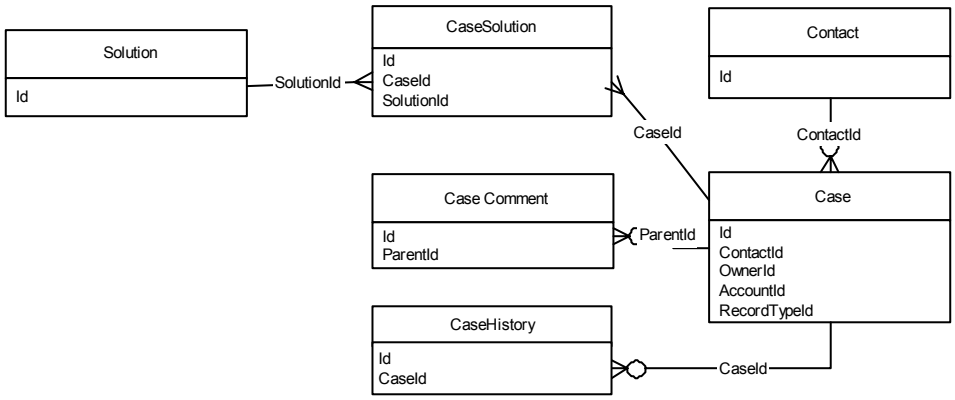
MAJOR OBJECTS



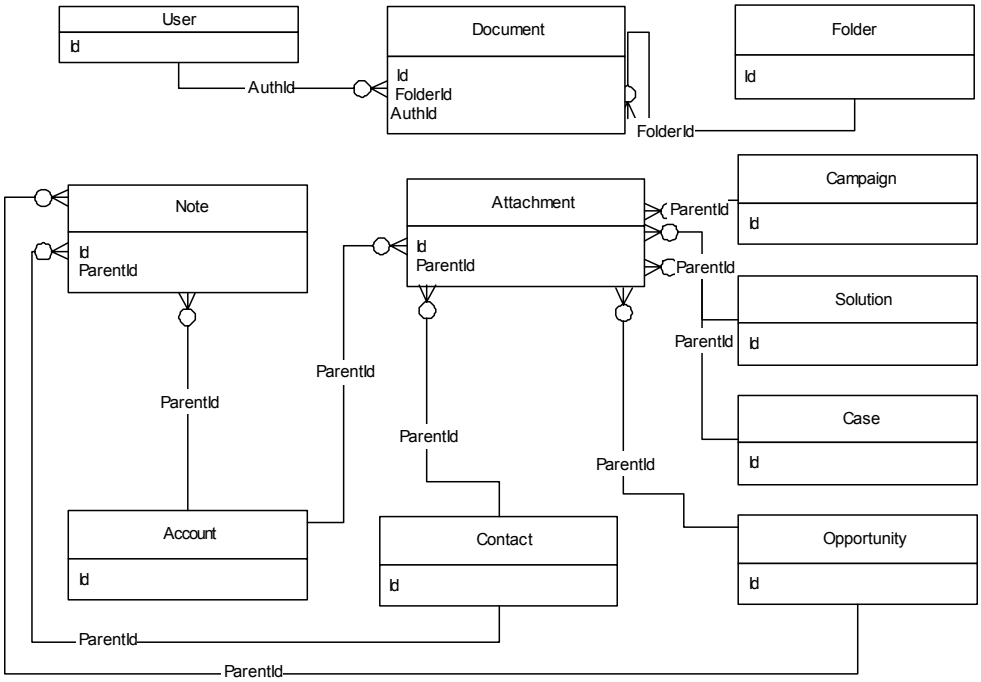
TASK AND EVENT OBJECTS



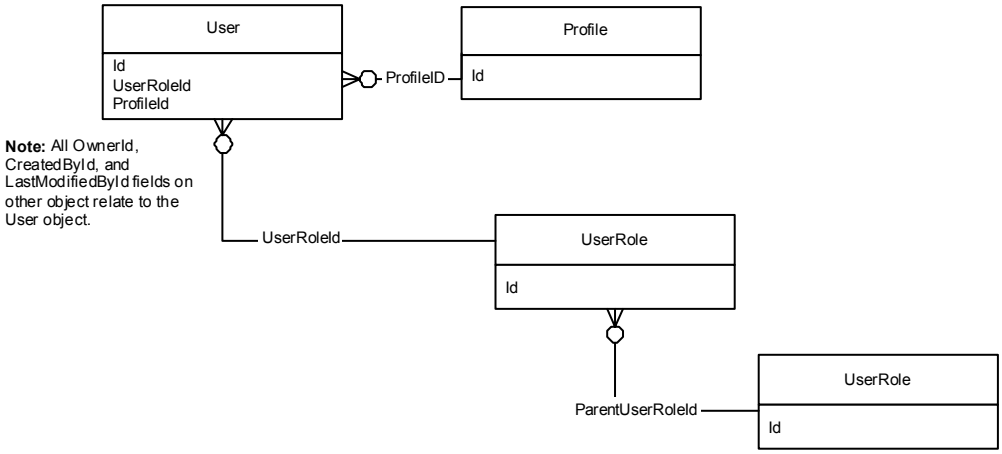
SUPPORT OBJECTS



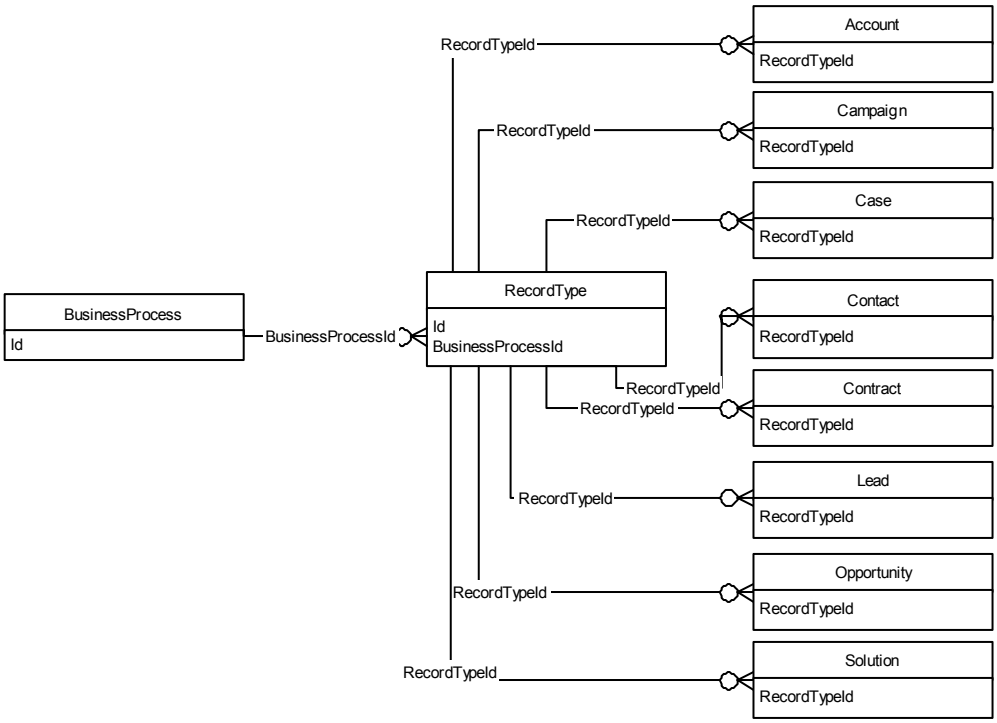
DOCUMENT, NOTE, AND ATTACHMENT OBJECTS



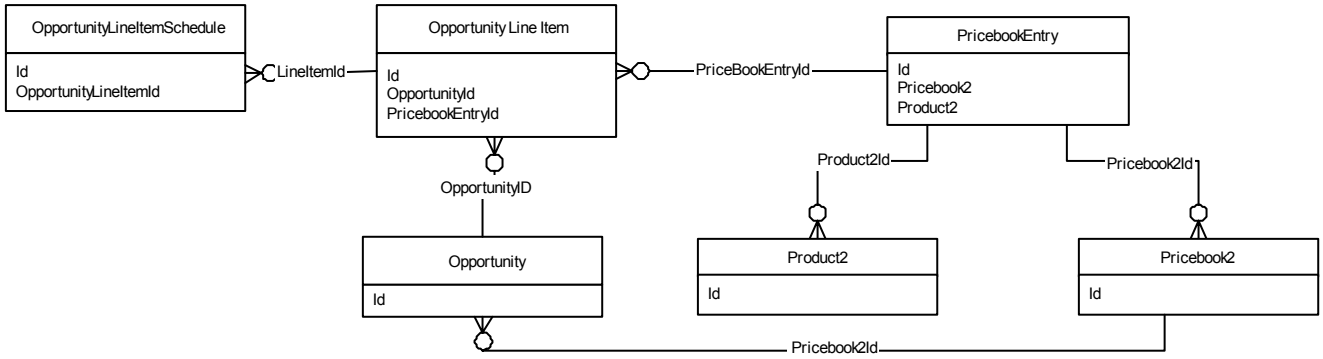
USER AND PROFILE OBJECTS



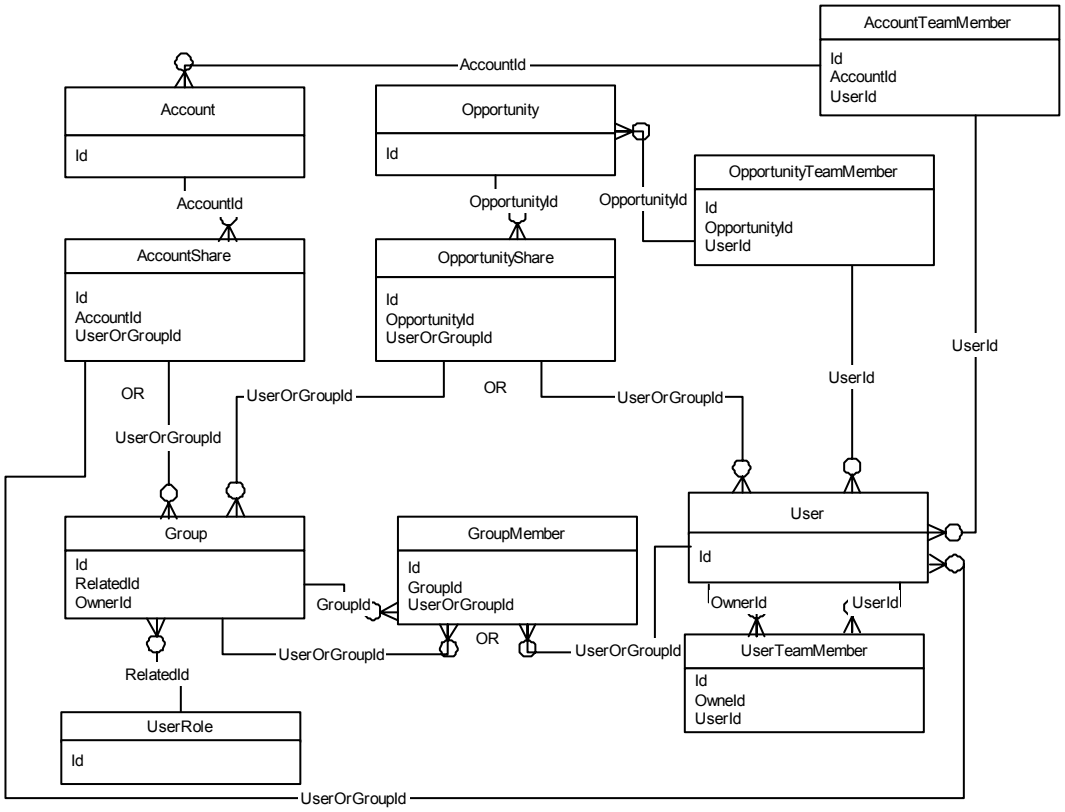
RECORD TYPE OBJECTS



PRODUCT AND SCHEDULE OBJECTS



SHARING AND TEAM SELLING OBJECTS



CHAPTER 7: sforce Partner Web Services API

This topic describes the sforce Partner Web services API. It contains the following sections:

INTRODUCING THE SFORCE PARTNER WEB SERVICES API

The sforce Web services API comes in two variations:

- **sforce Enterprise Web services API**—Used by enterprise developers to build client applications for a *single* salesforce.com organization.
- **sforce Partner Web services API**—Used for client applications that are metadata driven and dynamic in nature. It is particularly—but not exclusively—useful to salesforce.com partners who are building client applications for multiple organizations. As a loosely typed representation of the salesforce.com object model, it can be used to access data within any organization. It is more flexible, although not as easy to use, as its Enterprise counterpart.

This topic introduces the sforce Partner Web services API and describes how it differs from the sforce Enterprise Web services API. In general, the sforce Enterprise Web services API is more straightforward to use, while the sforce Partner Web services API is more flexible and dynamically adaptable to different organizations, allowing you to write a single application that can be used for multiple users and multiple organizations.

WSDL Files

To use the sforce Partner Web services API, you need a Partner WSDL file, which you can either:

- obtain from your organization's salesforce.com administrator, or
- generate in the **Setup** | **Integrate | WSDL Generator** area of the salesforce.com user interface according to the instructions in Step 2: Generate or Obtain the sforce Web Service WSDL For Your Organization on page 4.

While the enterprise.wsdl file (used with the sforce Enterprise Web services API) needs to be generated if custom fields or custom objects are added to an organization's salesforce.com information, the Partner WSDL file remains the same regardless of underlying changes in the organization's salesforce.com data.

API Calls in the sforce Partner Web Services API

The Partner WSDL file defines exactly the same API calls found in the enterprise.wsdl file. A client application using the Partner Web services API will likely use the following API calls to determine an organization's metadata:

Table 56: Object Metadata Calls in the sforce API

Task / Call	Description
describeGlobal	Retrieves a list of available objects for your organization's data.

Table 56: Object Metadata Calls in the sforce API (Continued)

Task / Call	Description
describeSObject	Describes metadata (field list and object properties) for the specified object.

To explore an organization's metadata, a client application can:

1. Call describeGlobal to obtain a list of available objects.
2. In the returned DescribeGlobalResult, retrieve an array of sObjects (`types` field).
3. Iterate through each sObject in the array, calling describeSObject to retrieve a list of fields and other properties for the sObject in the returned DescribeSObjectResult.

OBJECTS, FIELDS, AND FIELD DATA IN THE SFORCE PARTNER WEB SERVICES API

While the enterprise.wsdl file defines all of the specific objects (such as Account, Contact, and other objects described in sforce Objects on page 101) in a salesforce.com organization, the Partner WSDL file defines a single, generic object (sObject) that represents all of the objects. For a particular object, its type is defined in the `name` field in the returned DescribeSObjectResult.

In the sforce Partner Web services API, your client application code handles fields as arrays of name-value pairs that represent the field data. When referring to the names of individual fields, use the value in its `name` field of the Field type in the DescribeSObjectResult.

Languages vary in the way they handle name-value pairs and map typed values to the primitive XML data types defined in SOAP messages. In the sforce Enterprise Web services API, the mapping is handled implicitly. In the sforce Partner Web services API, however, you need to be more attentive to values and data types when building client applications. When specifying the value of a particular field, be sure to use a value that is valid for the field (range, format, and data type). Make sure that you understand the mapping between data types in your programming language with XML primitive data types (one of the values in the SOAPType field of the Field type in the DescribeSObjectResult).

QUERIES IN THE SFORCE PARTNER WEB SERVICES API

When using the query call in the sforce Partner Web services API, consider the following guidelines:

- The `queryString` parameter is case-insensitive. The sforce Web service will accept field names in the *`fieldList`* using any combination of uppercase and lowercase letters. However, in the QueryResult, the case of field names (both predefined and custom fields) will match exactly the value in the `name` field of the Field type in the DescribeSObjectResult. It is recommended that you use the proper case when specifying fields in the *`fieldList`*.
- For the Partner Web services API, the ordering of fields in the QueryResult is determined by the field order in the *`fieldList`*, not the field order in the WSDL file.
- The *`fieldList`* cannot contain duplicate field names. For example:
  - Invalid (returns an error): `"select firstname, lastname, firstname from User"`
  - Valid: `"select firstname, lastname from User"`
- The QueryResult always contains all of the fields specified in the *`fieldList`*, even if some of the fields contain no data (`null`). Although SOAP allows you to omit fields that contain no values in the result set, the sforce Web services API always returns an array containing all fields.

NAMESPACES IN THE SFORCE PARTNER WEB SERVICES API

In XML, every tag has a defined namespace. In the enterprise.wsdl, namespaces are handled implicitly. When using API calls in the sforce Partner Web services API, however, you need to explicitly specify the correct namespaces for sforce API calls, objects, and fields, and faults. This rule applies to predefined and custom objects and fields.

Table 57: Namespaces for the sforce Partner Web Services API

For	Namespace
API Calls	`partner.soap.sforce.com`
sObjects	`sobject.partner.soap.sforce.com`
Fields	`sobject.partner.soap.sforce.com`
Faults	`fault.partner.soap.sforce.com`

EXAMPLES

This section provides the following code samples in Java and C#:

- Sample query Calls
- Sample search Call
- Sample create Call
- Sample update Call

Sample query Calls

The following Java and C# examples show using the query call for the Partner API.

Java Example

```
private void querySample() {

  QueryResult qr = null;
  _QueryOptions qo = new _QueryOptions();
  qo.setBatchSize(new Integer(3));
  binding.setHeader(new
SforceServiceLocator().getServiceName().getNamespaceURI(), "QueryOptions", qo);

  qr = binding.query(
      "select id, Website, Name from Account where Name = 'Golden Straw'");
  if (qr.getSize() != 0) {
    SObject account = qr.getRecords()[0];
    System.out.println("Retrieved " + new Integer(qr.getSize()).toString() +
         " account(s) using Name = 'Golden Straw', ID = " +
         account.getId().getValue()+ ", website = " +
         account.get_any()[1].getValue());
  }

  qr = binding.query("select FirstName, LastName, Id from Contact");

  boolean bContinue = true;
  int loopCount = 0;
  while (bContinue) {
    System.out.println("Results set " + new Integer(loopCount++).toString() +
```

```
                              " - ");
    //process the query results
    for (int i = 0; i < qr.getRecords().length; i++) {
      SObject con = qr.getRecords()[i];
      org.apache.axis.message.MessageElement[] fields = con.get_any();
      String fName = fields[0].getValue();
      String lName = fields[1].getValue();
      if (fName == null) {
        System.out.println("Contact " + (i + 1) + ": " + lName);
      }
      else {
        System.out.println("Contact " + (i + 1) + ": " + fName + " " +
             lName);
      }
    }
    //handle the loop + 1 problem by checking to see if
    //the most recent queryResult
    if (qr.isDone()) {
      bContinue = false;
    }
    else {
      qr = binding.queryMore(qr.getQueryLocator());
    }
  }
}
```

C# Example

```
private void querySample()
{
  QueryResult qr = null;
  binding.QueryOptionsValue = new sforce.QueryOptions();
  binding.QueryOptionsValue.batchSize = 3;
  binding.QueryOptionsValue.batchSizeSpecified = true;

  qr = binding.query("select FirstName, LastName from Contact");

  bool bContinue = true;
  int loopCounter = 0;
  while (bContinue)
  {
    Console.WriteLine("\nResults Set " + Convert.ToString(loopCounter++) + " - ");
    //process the query results
    for (int i=0;i<qr.records.Length;i++)
    {
    sforce.sObject con = qr.records[i];
    string fName = con.Any[0].InnerText;
    string lName = con.Any[1].InnerText;
    if (fName == null)
      Console.WriteLine("Contact " + (i + 1) + ": " + lName);
    else
      Console.WriteLine("Contact " + (i + 1) + ": " + fName + " " + lName);
    }
    //handle the loop + 1 problem by checking to see if the most recent queryResult
    if (qr.done)
      bContinue = false;
    else
      qr = binding.queryMore(qr.queryLocator);
    }
```



```
    Console.WriteLine("\nQuery succesfully executed.");
    Console.Write("\nHit return to continue...");
    Console.ReadLine();
  }
}
```

Sample search Call

The following Java and C# examples show using the search call for the Partner API.

Java Example

```
private void searchSample() {

  SearchResult sr = null;
  QueryResult qr = null;
  sr = binding.search("find {4159017000} in phone fields returning
          contact(id, phone, firstname, lastname),
          lead(id, phone, firstname, lastname),
          account(id, phone, name)");
  SearchRecord[] records = sr.getSearchRecords();
  Vector contacts = new Vector();
  Vector leads = new Vector();
  Vector accounts = new Vector();

  if (records.length > 0) {
    for (int i=0;i<records.length;i++){
    SObject record = records[i].getRecord();
    if (record.getType().toLowerCase().equals("contact")) {
      contacts.add(record);
    } else if (record.getType().toLowerCase().equals("lead")){
      leads.add(record);
    } else if (record.getType().toLowerCase().equals("account")) {
      accounts.add(record);
    }
    }
    if (contacts.size() > 0) {
    System.out.println("Found " + new Integer(contacts.size()).toString() +
           " contacts:");
    for (int i=0;i<contacts.size();i++){
    SObject c = (SObject)contacts.get(i);
    System.out.println(c.getId().getValue() + " - " +
           c.get_any()[1].getValue() + " " +
           c.get_any()[2].getValue() + " - " + c.get_any()[0].getValue());
    }
  }
  if (leads.size() > 0) {
    System.out.println("Found " + new Integer(leads.size()).toString() +
         " leads:");
    for (int i=0;i<leads.size();i++){
    SObject l = (SObject) leads.get(i);
    System.out.println(l.getId().getValue() + " - " +
           l.get_any()[1].getValue() + " " +
           l.get_any()[2].getValue() + " - " + l.get_any()[0].getValue());
    }
  }
  if (accounts.size() > 0) {
    System.out.println("Found " + new Integer(accounts.size()).toString() +
         " accounts:");
```

```
    for (int i=0;i<accounts.size();i++){
    SObject a = (SObject) accounts.get(i);
    System.out.println(a.getId().getValue() + " - " +
           a.get_any()[1].getValue() + " - " + a.get_any()[0].getValue());
    }
  }
  } else {
    System.out.println("No records were found for the search.");
  }
}
```

C# Example

```
private void searchSample()
{
  sforce.SearchResult sr = null;
  sr = binding.search("find {4159017000} in phone fields returning " +
        "contact(id, phone, firstname, lastname), " +
        "lead(id, phone, firstname, lastname), " +
        "account(id, phone, name)");

  sforce.sObject[] records = sr.records;
  System.Collections.ArrayList contacts = new System.Collections.ArrayList();
  System.Collections.ArrayList leads = new System.Collections.ArrayList();
  System.Collections.ArrayList accounts = new System.Collections.ArrayList();

  if (sr.size > 0)
  {
    for (int i=0;i<records.Length;i++)
    {
      sforce.sObject record = records[i];
      if (record.type.ToLower().Equals("contact"))
      {
     contacts.Add(record);
      }
      else if (record.type.ToLower().Equals("lead"))
      {
        leads.Add(record);
      }
      else if (record.type.ToLower().Equals("account") )
      {
        accounts.Add(record);
      }
    }
    if (contacts.Count > 0)
    {
      Console.WriteLine("Found " + contacts.Count + " contacts:");
      for (int i=0;i<contacts.Count;i++)
      {
     sforce.sObject c = (sforce.sObject)contacts[i];
     Console.WriteLine(c.Any[2].InnerText + " " + c.Any[3].InnerText + " - " +
                c.Any[1].InnerText);
      }
    }
    if (leads.Count > 0)
    {
      Console.WriteLine("Found " + leads.Count + " leads:");
      for (int i=0;i<leads.Count;i++)
      {
```



```
     sforce.sObject l = (sforce.sObject)leads[i];
     Console.WriteLine(l.Any[2].InnerText + " " + l.Any[3].InnerText + " - " +
                l.Any[1].InnerText);
      }
    }
    if (accounts.Count > 0)
    {
      Console.WriteLine("Found " + accounts.Count + " accounts:");
      for (int i=0;i<accounts.Count;i++)
      {
     sforce.sObject a = (sforce.sObject)accounts[i];
     Console.WriteLine(a.Any[2].InnerText + " - " + a.Any[1].InnerText);
      }
    }
  }
  else
  {
    Console.WriteLine("No records were found for the search.");
  }
}
```

Sample create Call

The following Java and C# examples show using the create call for the Partner API.

Java Example

```
private void createContactSample() {
SObject[] cons = new SObject[1];
  MessageElement[] contact = new MessageElement[5];
  contact[0] = new MessageElement(new QName("FirstName"),"Joe");
  contact[1] = new MessageElement(new QName("LastName"),"Blow");
  contact[2] = new MessageElement(new QName("Salutation"),"Mr.");
  contact[3] = new MessageElement(new QName("Phone"),"999.999.9999");
  contact[4] = new MessageElement(new QName("Title"),"Purchasing Director");
  cons[0] = new SObject();
  cons[0].setType("Contact");
  cons[0].set_any(contact);

  SaveResult[] sr = binding.create(cons);
  for (int j = 0; j < sr.length; j++) {
    if (sr[j].isSuccess()) {
      System.out.println("A contact was created with an id of: "
                         + sr[j].getId().getValue());
    }
    else {
      //there were errors during the create call, go through the errors
      //array and write them to the screen
      for (int i = 0; i < sr[j].getErrors().length; i++) {
        //get the next error
        com.sforce.soap.partner.Error err = sr[j].getErrors()[i];
        System.out.println("Errors were found on item " +
                           new Integer(j).toString());
        System.out.println("Error code: " + err.getStatusCode().toString());
        System.out.println("Error message: " + err.getMessage());
      }
    }
  }
```



```
}
```

C# Example

```
private void createAccountSample()

  sforce.sObject account;
  sObject[] accs = new sObject[1];

  account = new sforce.sObject();
  System.Xml.XmlElement[] acct = new System.Xml.XmlElement[6];

  System.Xml.XmlDocument doc = new System.Xml.XmlDocument();

  acct[0] = doc.CreateElement("Industry"); acct[0].InnerText = "Farming";
  acct[1] = doc.CreateElement("Name"); acct[1].InnerText = "Golden Straw";
  acct[2] = doc.CreateElement("NumberOfEmployees"); acct[2].InnerText = "40";
  acct[3] = doc.CreateElement("Ownership"); acct[3].InnerText = "Privately Held";
  acct[4] = doc.CreateElement("Phone"); acct[4].InnerText = "666.666.6666";
  acct[5] = doc.CreateElement("Website"); acct[5].InnerText = "www.oz.com";
  account.type = "Account";
  account.Any = acct;
  accs[0] = account;

  //create the object(s) by sending the array to the web service
  SaveResult[] sr = binding.create(accs);
  for (int j=0;j<sr.Length;j++)
  {
    if (sr[j].success)
    {
    Console.Write(System.Environment.NewLine + "An account was create with an id
of: "
                + sr[j].id);
    }
    else
    {
    //there were errors during the create call, go through the errors
    //array and write them to the screen
    for (int i=0;i<sr[j].errors.Length;i++)
    {
      //get the next error
      Error err = sr[j].errors[i];
      Console.WriteLine("Errors were found on item " + j.ToString());
      Console.WriteLine("Error code is: " + err.statusCode.ToString());
      Console.WriteLine("Error message: " + err.message);
    }
    }
  }
}
```

Sample update Call

The following Java and C# examples show using the update call for the Partner API.

Java Example

```
private void updateAccountSample() {
```



```
  //create the account object to hold our changes
  SObject updateAccount = new SObject();
  updateAccount.setType("Account");
  //need to have the id so that web service knows which account to update
  updateAccount.setId(accounts[0]);
  //set a new value for the name property
  MessageElement[] ufields = new MessageElement[1];
  ufields[0] = new MessageElement(new QName("Name"),"New Account from Update
Sample");
  updateAccount.set_any(ufields);

  //create one that will throw an error
  SObject errorAccount = new SObject();
  errorAccount.setType("Account");
  errorAccount.setId(new ID("SLFKJLFKJ"));
  errorAccount.setFieldsToNull(new String[] {"Name"});

  //call the update passing an array of object
  SaveResult[] saveResults = binding.update(new SObject[] {updateAccount,
                                                    errorAccount});
  //loop through the results, checking for errors
  for (int j = 0; j < saveResults.length; j++) {
    System.out.println("Item: " + new Integer(j).toString());
    if (saveResults[j].isSuccess()) {
      System.out.println("An account with an id of: " +
           saveResults[j].getId().getValue() + " was updated.\n");
    }
    else {
      System.out.println("Item " + new Integer(j).toString() +
                         " had an error updating.");
      System.out.println("    The error reported was: " +
           saveResults[j].getErrors()[0].getMessage() + "\n");
    }
  }
}
```

C# Example

```
private void updateAccountSample()
{
  //create the account object to hold our changes
  sforce.sObject updateAccount = new sforce.sObject();
  //need to have the id so that web service knows which account to update

  //set a new value for the name property
  updateAccount.Id = "00130000001dmJT";
  System.Xml.XmlDocument doc = new System.Xml.XmlDocument();
  System.Xml.XmlElement nameElement = new System.Xml.XmlElement("Name");
  nameElement.InnerText = "New Account Name from Update Sample";

  updateAccount.Any = new System.Xml.XmlElement[] { nameElement };
  updateAccount.type = "Account";

  //call the update passing an array of object
  SaveResult[] saveResults = binding.update(new sforce.sObject[] { updateAccount
});

  //loop through the results, checking for errors
  for (int j=0;j<saveResults.Length;j++)
```



```
  {
    Console.WriteLine("Item: " + j);
    if (saveResults[j].success)
      Console.WriteLine("An account with an id of: " + saveResults[j].id +
            " was updated.\n");
    else
    {
    Console.WriteLine("Item " + j.ToString() + " had an error updating.");
    Console.WriteLine("    The error reported was: " +
             saveResults[j].errors[0].message + "\n");
    }
  }
}
```


CHAPTER 8: SOAP Header Options

This topic describes the following SOAP header options available to client applications that use the sforce Web services API. All of these options are available in both the Enterprise and Partner WSDL files.

Table 58: Header Options in the sforce API

Task / Call	Description
AssignmentRuleHeader	Specifies the assignment rule to use when creating or updating a Case or Lead.
QueryOptions	Sets the batch size for query results.
SaveOptions (Deprecated)	Specifies the assignment rule to use when creating or updating a Case or Lead.
SessionHeader	Specifies the session ID returned from the sforce server after a successful login.

AssignmentRuleHeader

Specifies the assignment rule to use when creating or updating a Case or Lead. If the **`AssignmentRuleHeader`** is not specified in a create or update call, then no assignment rule is applied.

API Calls

create, update

Fields

Table 59: AssignmentRuleHeader Fields

Element Name	Type	Description
`assignmentRuleId`	ID	ID of a specific assignment rule to run for the Case or Lead. Can be an inactive assignment rule. The ID can be retrieved by querying the AssignmentRule object (for details, see AssignmentRule on page 122). If specified, do not specify **`useDefaultRule`**.
`useDefaultRule`	boolean	If True, uses the default (active) assignment rule for a Case or Lead. If specified, do not specify an **`assignmentRuleId`**.

Sample Code

For a code example, see Lead on page 141.

See Also

create on page 49
update on page 90
AssignmentRule on page 122

QueryOptions

Specifies the batch size for queries.

Associated API Calls

query, queryMore

Field

Table 60: QueryOptions Field

Element Name	Type	Description
`batchSize`	int	Batch size for the number of records returned in a query or queryMore call. Default is 2000.

Sample Code

For code examples, see Changing the Batch Size in Queries on page 32.

See Also

query on page 79
Changing the Batch Size in Queries on page 32

SaveOptions (Deprecated)

Specifies the assignment rule to use when creating or updating a Case or Lead. If `SaveOptions` is not specified in a create or update call, then no assignment rule is applied.

Note It is recommended that you use the AssignmentRuleHeader option instead.

API Calls

create, update

Fields

Table 61: SaveOptions Fields

Element Name	Type	Description
autoAssign	boolean	If True, run an assignment rule. If False (the default), no assignment rule is run (regardless of the value of the **assignmentRuleId** element).
assignmentRuleId	ID	ID of a specific assignment rule to run (can be an inactive rule). If blank, then the default assignment rule is run (in both cases, **autoAssign** must be True).

Sample Java Code

```
//Create the save options header and add it to the proxy binding
_SaveOptions so = new _SaveOptions();
// To obtain the assignment rule ID, query the AssignmentRule object,
// iterate through the results, and get the ID of the assignment rule
// that you want to use.
ID arID = new ID();
arID.setValue("01Q3000000006Ra");
so.setAssignmentRuleId(arID);
//Be sure to set the setAutoAssign flag to true
so.setAutoAssign(true);
binding.setHeader(new SforceServiceLocator().getServiceName().getNamespaceURI(),
"SaveOptions", so);
```

See Also

AssignmentRuleHeader on page 196
create on page 49
update on page 90
AssignmentRule on page 122

SessionHeader

Specifies the session ID returned from the sforce server after a successful login. This session ID is used in all subsequent calls.

Field

Table 62: SessionHeader

Element Name	Type	Description
sessionId	string	Session ID returned by the login call to be used for subsequent call authentication.

Sample Code

For sample code, see the examples provided for login on page 76.

See Also

login on page 76

CHAPTER 9: Sample SOAP Messages

This topic provides sample input and output SOAP messages for the sforce Web services API calls. It includes the following topics:

For detailed information about the sforce API calls, see the following topics:

- sforce API Calls on page 22
- sforce Utility API Calls on page 94

Sample SOAP Messages—convertLead

This topic provides the following sample SOAP messages:

Sample Request Message—convertLead Call—Enterprise API

```
POST /services/Soap/c/5.0 HTTP/1.0Content-Type: text/xml; charset=utf-8Accept:
application/soap+xml, application/dime, multipart/related, text/*User-Agent:
Axis/1.1Host: na1-api.salesforce.comCache-Control: no-cachePragma: no-
cacheSOAPAction: ""Content-Length: 1103<?xml version="1.0" encoding="UTF-8"?>
<soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
xmlns:xsd="http://www.w3.org/2001/XMLSchema" xmlns:xsi="http://www.w3.org/2001/
XMLSchema-instance">
```

```
 <soapenv:Header>
  <ns1:SessionHeader soapenv:mustUnderstand="0"
xmlns:ns1="urn:enterprise.soap.sforce.com">
   <ns1:sessionId>6l9auFNpYPGRiIh9St82kkGBlUtIlWMt0_jXlNKOSU</ns1:sessionId>
  </ns1:SessionHeader>
 </soapenv:Header>
 <soapenv:Body>
  <convertLead xmlns="urn:enterprise.soap.sforce.com">
   <leadConverts>
    <accountId>001300000019ykD</accountId>
    <contactId>00330000001obW4</contactId>
    <convertedStatus>Closed - Converted</convertedStatus>
    <doNotCreateOpportunity>false</doNotCreateOpportunity>
    <leadId>00Q30000000zwXA</leadId>
    <opportunityName>Converted Lead Opportunity</opportunityName>
    <overwriteLeadSource>true</overwriteLeadSource>
    <ownerId xsi:nil="true"/>
    <sendNotificationEmail>true</sendNotificationEmail>
   </leadConverts>
  </convertLead>
 </soapenv:Body>
</soapenv:Envelope>
```

Sample Response Message—convertLead Call—Enterprise API

```
HTTP/1.0 200 OKServer: sfdcContent-Type: text/xml; charset=utf-8Date: Mon, 08 Nov
2004 20:22:09 GMT<?xml version="1.0" encoding="UTF-8"?>
<soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
xmlns:xsd="http://www.w3.org/2001/XMLSchema" xmlns:xsi="http://www.w3.org/2001/
XMLSchema-instance">
 <soapenv:Body>
  <convertLeadResponse xmlns="urn:enterprise.soap.sforce.com">
   <result>
    <accountId>001300000019ykDAAQ</accountId>
    <contactId>00330000001obW4AAI</contactId>
    <errors xsi:nil="true"/>
    <leadId>00Q30000000zwXAEAY</leadId>
    <opportunityId>006D0000001TjzSIAS</opportunityId>
    <success>true</success>
   </result>
  </convertLeadResponse>
 </soapenv:Body>
</soapenv:Envelope>
```

Sample Request Message—convertLead Call—Partner API

```
POST /services/Soap/c/5.0 HTTP/1.0
Content-Type: text/xml; charset=utf-8
Accept: application/soap+xml, application/dime, multipart/related, text/*
User-Agent: Axis/1.1
Host: na1-api.salesforce.com
Cache-Control: no-cache
Pragma: no-cache
SOAPAction: ""
Content-Length: 1103

<?xml version="1.0" encoding="UTF-8"?>
```



```
<soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
xmlns:xsd="http://www.w3.org/2001/XMLSchema" xmlns:xsi="http://www.w3.org/2001/
XMLSchema-instance">
 <soapenv:Header>
  <ns1:SessionHeader soapenv:mustUnderstand="0"
xmlns:ns1="urn:partner.soap.sforce.com">
   <ns1:sessionId>QwWsHJyTPW.1pd0_jXlNKOSU</ns1:sessionId>
  </ns1:SessionHeader>
 </soapenv:Header>
 <soapenv:Body>
  <convertLead xmlns="urn:partner.soap.sforce.com">
   <leadConverts>
    <accountId>001300000019ykI</accountId>
    <contactId>00330000001obWD</contactId>
    <convertedStatus>Closed - Converted</convertedStatus>
    <doNotCreateOpportunity>false</doNotCreateOpportunity>
    <leadId>00Q30000000zwXC</leadId>
    <opportunityName>Partner Opp</opportunityName>
    <overwriteLeadSource>true</overwriteLeadSource>
    <ownerId xsi:nil="true"/>
    <sendNotificationEmail>true</sendNotificationEmail>
   </leadConverts>
  </convertLead>
 </soapenv:Body>
</soapenv:Envelope>
```

Sample Response Message—convertLead Call—Partner API

```
HTTP/1.0 200 OK
Server: sfdc
Content-Type: text/xml; charset=utf-8
Date: Mon, 08 Nov 2004 20:22:09 GMT

<?xml version="1.0" encoding="UTF-8"?>
<soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
xmlns:xsd="http://www.w3.org/2001/XMLSchema" xmlns:xsi="http://www.w3.org/2001/
XMLSchema-instance">
 <soapenv:Body>
  <convertLeadResponse xmlns="urn:partner.soap.sforce.com">
   <result>
    <accountId>001300000019ykIAAQ</accountId>
    <contactId>00330000001obWDAAY</contactId>
    <errors xsi:nil="true"/>
    <leadId>00Q30000000zwXCEAY</leadId>
    <opportunityId>006D0000001TjzUIAS</opportunityId>
    <success>true</success>
   </result>
  </convertLeadResponse>
 </soapenv:Body>
</soapenv:Envelope>
```

Sample SOAP Messages—create

This topic provides the following sample SOAP messages:

- Sample Request Message—create Call—Enterprise API
- Sample Response Message—create Call—Enterprise API

Sample Request Message—create Call—Enterprise API

```
POST /services/Soap/c/5.0 HTTP/1.0
Content-Type: text/xml; charset=utf-8
Accept: application/soap+xml, application/dime, multipart/related, text/*
User-Agent: Axis/1.1
Host: na1.salesforce.com
Cache-Control: no-cache
Pragma: no-cache
SOAPAction: ""
Content-Length: 2200

<?xml version="1.0" encoding="UTF-8"?>
   <soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
xmlns:xsd="http://www.w3.org/2001/XMLSchema" xmlns:xsi="http://www.w3.org/2001/
XMLSchema-instance">
      <soapenv:Header>
         <ns1:SessionHeader soapenv:mustUnderstand="0"
xmlns:ns1="urn:enterprise.soap.sforce.com">
            <ns2:sessionId
xmlns:ns2="urn:enterprise.soap.sforce.com">WgMUbbkPm_jXlNKOSU</ns2:sessionId>
         </ns1:SessionHeader>
      </soapenv:Header>
      <soapenv:Body>
         <create xmlns="urn:enterprise.soap.sforce.com">
            <sObjects xsi:type="ns3:Account"
xmlns:ns3="urn:sobject.enterprise.soap.sforce.com">
               <ns3:AccountNumber>0000000</ns3:AccountNumber>
               <ns3:BillingCity>Wichita</ns3:BillingCity>
               <ns3:BillingCountry>US</ns3:BillingCountry>
               <ns3:BillingPostalCode>87901</ns3:BillingPostalCode>
               <ns3:BillingState>KA</ns3:BillingState>
               <ns3:BillingStreet>4322 Haystack Boulevard</ns3:BillingStreet>
               <ns3:Description>World class hay makers.</ns3:Description>
               <ns3:Fax>555.555.5555</ns3:Fax>
               <ns3:Industry>Farming</ns3:Industry>
               <ns3:Name>Golden Straw</ns3:Name>
               <ns3:NumberOfEmployees>40</ns3:NumberOfEmployees>
               <ns3:Ownership>Privately Held</ns3:Ownership>
               <ns3:Phone>666.666.6666</ns3:Phone>
               <ns3:Website>www.oz.com</ns3:Website>
            </sObjects>
         </create>
      </soapenv:Body>
   </soapenv:Envelope>
```

Sample Response Message—create Call—Enterprise API

```
HTTP/1.0 200 OK
Server: sfdc
Content-Type: text/xml; charset=utf-8
Date: Wed, 07 Jul 2004 18:01:21 GMT

<?xml version="1.0" encoding="UTF-8"?>
```

```
   <soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
xmlns:xsd="http://www.w3.org/2001/XMLSchema" xmlns:xsi="http://www.w3.org/2001/
XMLSchema-instance">
      <soapenv:Body>
         <createResponse xmlns="urn:enterprise.soap.sforce.com">
            <result>
               <errors xsi:nil="true"/>
               <id>00130000002K9TnAAK</id>
               <success>true</success>
            </result>
         </createResponse>
      </soapenv:Body>
   </soapenv:Envelope>
```

Sample Request Message—create Call—Partner API

```
POST https://na1.salesforce.com/services/Soap/u/5.0 HTTP/1.1
User-Agent: Mozilla/4.0 (compatible; MSIE 6.0; MS Web Services Client Protocol
1.1.4322.573)
Content-Type: text/xml; charset=utf-8
SOAPAction: ""
Content-Length: 2056
Expect: 100-continue
Host: na1.salesforce.com

<?xml version="1.0" encoding="utf-8"?>
   <soap:Envelope xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xmlns:xsd="http://
www.w3.org/2001/XMLSchema">
      <soap:Header>
         <SessionHeader xmlns="urn:partner.soap.sforce.com">
            <sessionId>4I1yNabX__jXlNKOSU</sessionId>
         </SessionHeader>
      </soap:Header>
      <soap:Body>
         <create xmlns="urn:partner.soap.sforce.com">
            <sObjects>
               <type xmlns="urn:sobject.partner.soap.sforce.com">Account</type>
               <Id xsi:nil="true" xmlns="urn:sobject.partner.soap.sforce.com" />
               <AccountNumber xmlns="">0000000</AccountNumber>
               <BillingCity xmlns="">Wichita</BillingCity>
               <BillingCountry xmlns="">US</BillingCountry>
               <BillingState xmlns="">KA</BillingState>
               <BillingStreet xmlns="">4322 Haystack Boulevard</BillingStreet>
               <BillingPostalCode xmlns="">87901</BillingPostalCode>
               <Description xmlns="">World class hay makers.</Description>
               <Fax xmlns="">555.555.5555</Fax>
               <Industry xmlns="">Farming</Industry>
               <Name xmlns="">Golden Straw</Name>
               <NumberOfEmployees xmlns="">40</NumberOfEmployees>
               <Ownership xmlns="">Privately Held</Ownership>
               <Phone xmlns="">666.666.6666</Phone>
               <Website xmlns="">www.oz.com</Website>
            </sObjects>
            <sObjects>
               <type xmlns="urn:sobject.partner.soap.sforce.com">Account</type>
               <Id xsi:nil="true" xmlns="urn:sobject.partner.soap.sforce.com" />
               <AccountNumber xmlns="">0000000</AccountNumber>
               <BillingCity xmlns="">Wichita</BillingCity>
```



```
                <BillingCountry xmlns="">US</BillingCountry>
                <BillingState xmlns="">KA</BillingState>
                <BillingStreet xmlns="">4322 Haystack Boulevard</BillingStreet>
                <BillingPostalCode xmlns="">87901</BillingPostalCode>
                <Description xmlns="">World class hay makers.</Description>
                <Fax xmlns="">555.555.5555</Fax>
                <Industry xmlns="">Farming</Industry>
                <Name xmlns="">Golden Straw</Name>
                <NumberOfEmployees xmlns="">40</NumberOfEmployees>
                <Ownership xmlns="">Privately Held</Ownership>
                <Phone xmlns="">666.666.6666</Phone>
                <Website xmlns="">www.oz.com</Website>
             </sObjects>
          </create>
       </soap:Body>
    </soap:Envelope>
```

Sample Response Message—create Call—Partner API

```
HTTP/1.1 200 OK
Server: sfdc
Content-Type: text/xml; charset=utf-8
Transfer-Encoding: chunked
Date: Wed, 07 Jul 2004 16:39:39 GMT

<?xml version="1.0" encoding="UTF-8"?>
   <soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
xmlns:xsd="http://www.w3.org/2001/XMLSchema" xmlns:xsi="http://www.w3.org/2001/
XMLSchema-instance">
      <soapenv:Body>
         <createResponse xmlns="urn:partner.soap.sforce.com">
            <result>
               <errors xsi:nil="true"/>
               <id>00130000002K9QgAAK</id>
               <success>true</success>
            </result>
            <result>
               <errors xsi:nil="true"/>
               <id>00130000002K9QhAAK</id>
               <success>true</success>
            </result>
         </createResponse>
      </soapenv:Body>
   </soapenv:Envelope>
```

Sample SOAP Messages—delete

This topic provides the following sample SOAP messages for the delete call:

- Sample Request Message—delete Call—Enterprise API
- Sample Response Message—delete Call—Enterprise API
- Sample Request Message—delete Call—Partner API
- Sample Response Message—delete Call—Partner API

Sample Request Message—delete Call—Enterprise API

```
POST /services/Soap/c/5.0 HTTP/1.0
Content-Type: text/xml; charset=utf-8
Accept: application/soap+xml, application/dime, multipart/related, text/*
User-Agent: Axis/1.1
Host: na1.salesforce.com
Cache-Control: no-cache
Pragma: no-cache
SOAPAction: ""
Content-Length: 704

<?xml version="1.0" encoding="UTF-8"?>
   <soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
xmlns:xsd="http://www.w3.org/2001/XMLSchema" xmlns:xsi="http://www.w3.org/2001/
XMLSchema-instance">
      <soapenv:Header>
         <ns1:SessionHeader soapenv:mustUnderstand="0"
xmlns:ns1="urn:enterprise.soap.sforce.com">
            <ns2:sessionId
xmlns:ns2="urn:enterprise.soap.sforce.com">uejMEgecf8yidA3zCFeT6baks1vjXlNKOSU</
ns2:sessionId>
         </ns1:SessionHeader>
      </soapenv:Header>
      <soapenv:Body>
         <delete xmlns="urn:enterprise.soap.sforce.com">
            <ids>00130000002K9TpAAK</ids>
            <ids>00130000002K9TqAAK</ids>
         </delete>
      </soapenv:Body>
   </soapenv:Envelope>
```

Sample Response Message—delete Call—Enterprise API

```
HTTP/1.0 200 OK
Server: sfdc
Content-Type: text/xml; charset=utf-8
Date: Wed, 07 Jul 2004 18:13:11 GMT

<?xml version="1.0" encoding="UTF-8"?>
   <soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
xmlns:xsd="http://www.w3.org/2001/XMLSchema" xmlns:xsi="http://www.w3.org/2001/
XMLSchema-instance">
      <soapenv:Body>
         <deleteResponse xmlns="urn:enterprise.soap.sforce.com">
            <result>
               <errors xsi:nil="true"/>
               <id>00130000002K9TpAAK</id>
               <success>true</success>
            </result>
            <result>
               <errors xsi:nil="true"/>
               <id>00130000002K9TqAAK</id>
               <success>true</success>
            </result>
         </deleteResponse>
      </soapenv:Body>
   </soapenv:Envelope>
```

Sample Request Message—delete Call—Partner API

```
POST https://na1.salesforce.com/services/Soap/u/5.0 HTTP/1.1
User-Agent: Mozilla/4.0 (compatible; MSIE 6.0; MS Web Services Client Protocol
1.1.4322.573)
Content-Type: text/xml; charset=utf-8
SOAPAction: ""
Content-Length: 570
Expect: 100-continue
Host: na1.salesforce.com
            <?xml version="1.0" encoding="utf-8"?>
               <soap:Envelope xmlns:soap="http://schemas.xmlsoap.org/soap/
envelope/" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns:xsd="http://www.w3.org/2001/XMLSchema">
                  <soap:Header>
                     <SessionHeader xmlns="urn:partner.soap.sforce.com">
                      <sessionId>4I1yNabX_5FyOGOnuFO64VpRP0791eQNkq3Vi_jXlNKOSU</
sessionId>
                     </SessionHeader>
                  </soap:Header>
                  <soap:Body>
                     <delete xmlns="urn:partner.soap.sforce.com">
                        <ids>00130000002K9QgAAK</ids>
                        <ids>00130000002K9QhAAK</ids>
                     </delete>
                  </soap:Body>
               </soap:Envelope>
```

Sample Response Message—delete Call—Partner API

```
HTTP/1.1 200 OK
Server: sfdc
Content-Type: text/xml; charset=utf-8
Transfer-Encoding: chunked
Date: Wed, 07 Jul 2004 16:58:58 GMT

<?xml version="1.0" encoding="UTF-8"?>
<soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
xmlns:xsd="http://www.w3.org/2001/XMLSchema" xmlns:xsi="http://www.w3.org/2001/
XMLSchema-instance">
      <soapenv:Body>
            <deleteResponse xmlns="urn:partner.soap.sforce.com">
                        <result>
                           <errors xsi:nil="true"/>
                           <id>00130000002K9QgAAK</id>
                           <success>true</success>
                        </result>
                        <result>
                           <errors xsi:nil="true"/>
                           <id>00130000002K9QhAAK</id>
                           <success>true</success>
                        </result>
                     </deleteResponse>
                  </soapenv:Body>
               </soapenv:Envelope>
```


Sample SOAP Messages—describeGlobal

This topic provides the following sample SOAP messages for the describeGlobal call:

- Sample Request Message—describeGlobal Call—Enterprise API
- Sample Response Message—describeGlobal Call—Enterprise API
- Sample Request Message—describeGlobal Call—Partner API
- Sample Response Message—describeGlobal Call—Partner API

Sample Request Message—describeGlobal Call—Enterprise API

```
POST /services/Soap/c/5.0 HTTP/1.0
Content-Type: text/xml; charset=utf-8
Accept: application/soap+xml, application/dime, multipart/related, text/*
User-Agent: Axis/1.1
Host: na1.salesforce.com
Cache-Control: no-cache
Pragma: no-cache
SOAPAction: ""
Content-Length: 635

<?xml version="1.0" encoding="UTF-8"?>
   <soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
xmlns:xsd="http://www.w3.org/2001/XMLSchema" xmlns:xsi="http://www.w3.org/2001/
XMLSchema-instance">
      <soapenv:Header>
         <ns1:SessionHeader soapenv:mustUnderstand="0"
xmlns:ns1="urn:enterprise.soap.sforce.com">
            <ns2:sessionId
xmlns:ns2="urn:enterprise.soap.sforce.com">uejMEgecf8yidA3zCFeT6baks1vD</
ns2:sessionId>
         </ns1:SessionHeader>
      </soapenv:Header>
      <soapenv:Body>
         <describeGlobal xmlns="urn:enterprise.soap.sforce.com"/>
      </soapenv:Body>
   </soapenv:Envelope>
```

Sample Response Message—describeGlobal Call—Enterprise API

```
HTTP/1.0 200 OK
Server: sfdc
Content-Type: text/xml; charset=utf-8
Date: Wed, 07 Jul 2004 18:09:09 GMT

<?xml version="1.0" encoding="UTF-8"?>
   <soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
xmlns:xsd="http://www.w3.org/2001/XMLSchema" xmlns:xsi="http://www.w3.org/2001/
XMLSchema-instance">
      <soapenv:Body>
         <describeGlobalResponse xmlns="urn:enterprise.soap.sforce.com">
            <result>
               <encoding>UTF-8</encoding>
               <maxBatchSize>500</maxBatchSize>
               <types>Account</types>
               <types>AccountContactRole</types>
               <types>AccountShare</types>
```

```
                <types>Approval</types>
                <types>AssignmentRule</types>
                <types>Attachment</types>
                <types>BusinessHours</types>
                <types>BusinessProcess</types>
                <types>Campaign</types>
                ...
                ...
                ...
                <types>Task</types>
                <types>TaskPriority</types>
                <types>TaskStatus</types>
                <types>User</types>
                <types>UserRole</types>
                <types>UserTeamMember</types>
                <types>WebLink</types>
             </result>
          </describeGlobalResponse>
       </soapenv:Body>
    </soapenv:Envelope>
```

Sample Request Message—describeGlobal Call—Partner API

```
POST https://na1.salesforce.com/services/Soap/u/5.0 HTTP/1.1
User-Agent: Mozilla/4.0 (compatible; MSIE 6.0; MS Web Services Client Protocol
1.1.4322.573)
Content-Type: text/xml; charset=utf-8
SOAPAction: ""
Content-Length: 513
Expect: 100-continue
Host: na1.salesforce.com

<?xml version="1.0" encoding="utf-8"?>
   <soap:Envelope xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xmlns:xsd="http://
www.w3.org/2001/XMLSchema">
      <soap:Header>
         <SessionHeader xmlns="urn:partner.soap.sforce.com">
            <sessionId>4I1yNabX_5FyOGOnu_jXlNKOSU</sessionId>
         </SessionHeader>
      </soap:Header>
      <soap:Body>
         <describeGlobal xmlns="urn:partner.soap.sforce.com" />
      </soap:Body>
   </soap:Envelope>
```

Sample Response Message—describeGlobal Call—Partner API

```
HTTP/1.1 200 OK
Server: sfdc
Content-Type: text/xml; charset=utf-8
Transfer-Encoding: chunked
Date: Wed, 07 Jul 2004 16:50:28 GMT

<?xml version="1.0" encoding="UTF-8"?>
```



```
   <soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
xmlns:xsd="http://www.w3.org/2001/XMLSchema" xmlns:xsi="http://www.w3.org/2001/
XMLSchema-instance">
      <soapenv:Body>
         <describeGlobalResponse xmlns="urn:partner.soap.sforce.com">
            <result>
               <encoding>UTF-8</encoding>
               <maxBatchSize>500</maxBatchSize>
               <types>Account</types>
               <types>AccountContactRole</types>
               <types>AccountShare</types>
               <types>Approval</types>
               <types>AssignmentRule</types>
               ...
               ...
               ...
               <types>TaskStatus</types>
               <types>User</types>
               <types>UserRole</types>
               <types>UserTeamMember</types>
               <types>WebLink</types>
            </result>
         </describeGlobalResponse>
      </soapenv:Body>
   </soapenv:Envelope>
```

Sample SOAP Messages—describeLayout

This topic provides the following sample SOAP messages:

- Sample Request Message—describeLayout Call—Enterprise API
- Sample Response Message—describeLayout Call—Enterprise API
- Sample Request Message—describeLayout Call—Partner API
- Sample Response Message—describeLayout Call—Partner API

Sample Request Message—describeLayout Call—Enterprise API

```
POST /services/Soap/u/5.0 HTTP/1.0
Content-Type: text/xml; charset=utf-8
Accept: application/soap+xml, application/dime, multipart/related, text/*
User-Agent: Axis/1.1
Host: www.salesforce.com
Cache-Control: no-cache
Pragma: no-cache
SOAPAction: ""
Content-Length: 660

<?xml version="1.0" encoding="UTF-8"?>
<soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
xmlns:xsd="http://www.w3.org/2001/XMLSchema" xmlns:xsi="http://www.w3.org/2001/
XMLSchema-instance">
 <soapenv:Header>
  <ns1:SessionHeader soapenv:mustUnderstand="0"
xmlns:ns1="urn:enterprise.soap.sforce.com">
   <ns1:sessionId>Y0PQJaMRQGBlUtIlWMt0_jXlNKOSU</ns1:sessionId>
  </ns1:SessionHeader>
 </soapenv:Header>
```



```
 <soapenv:Body>
  <describeLayout xmlns="urn:enterprise.soap.sforce.com">
   <sObjectType>Lead</sObjectType>
  </describeLayout>
 </soapenv:Body>
</soapenv:Envelope>
```

Sample Response Message—describeLayout Call—Enterprise API

See Sample Response Message—describeLayout Call—Partner API on page 211.

Sample Request Message—describeLayout Call—Partner API

```
POST /services/Soap/u/5.0 HTTP/1.0
Content-Type: text/xml; charset=utf-8
Accept: application/soap+xml, application/dime, multipart/related, text/*
User-Agent: Axis/1.1
Host: www.salesforce.com
Cache-Control: no-cache
Pragma: no-cache
SOAPAction: ""
Content-Length: 660

<?xml version="1.0" encoding="UTF-8"?>
<soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
xmlns:xsd="http://www.w3.org/2001/XMLSchema" xmlns:xsi="http://www.w3.org/2001/
XMLSchema-instance">
 <soapenv:Header>
  <ns1:SessionHeader soapenv:mustUnderstand="0"
xmlns:ns1="urn:partner.soap.sforce.com">
  <ns1:sessionId>XgmiShQdysIHGb2rlGWn_RNTHhV1xAfr38Fnd0_jXlNKOSU</ns1:sessionId>
  </ns1:SessionHeader>
 </soapenv:Header>
 <soapenv:Body>
  <describeLayout xmlns="urn:partner.soap.sforce.com">
   <sObjectType>Account</sObjectType>
  </describeLayout>
 </soapenv:Body>
</soapenv:Envelope>
```

Sample Response Message—describeLayout Call—Partner API

```
HTTP/1.0 200 OK
Server: sfdc
Content-Type: text/xml; charset=utf-8
Date: Mon, 08 Nov 2004 23:27:27 GMT

<?xml version="1.0" encoding="UTF-8"?>
   <soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
xmlns:xsd="http://www.w3.org/2001/XMLSchema" xmlns:xsi="http://www.w3.org/2001/
XMLSchema-instance">
      <soapenv:Body>
         <describeLayoutResponse xmlns="urn:partner.soap.sforce.com">
            <result>
               <layouts>
                  <detailLayoutSections>
                     <columns>2</columns>
```



```
                        <heading>Account Information</heading>
                        <layoutRows>
                           <layoutItems>
                              <editable>false</editable>
                              <label>Account Owner</label>
                              <layoutComponents>
                                 <tabOrder>1</tabOrder>
                                 <type>Field</type>
                                 <value>OwnerId</value>
                              </layoutComponents>
                              <placeholder>false</placeholder>
                              <required>false</required>
                           </layoutItems>
                           <layoutItems>
                              <editable>false</editable>
                              <label>Rating</label>
                              <layoutComponents>
                                 <tabOrder>9</tabOrder>
                                 <type>Field</type>
                                 <value>Rating</value>
                              </layoutComponents>
                              <placeholder>false</placeholder>
                              <required>false</required>
                           </layoutItems>
                           <numItems>2</numItems>
                        </layoutRows>

...
               </layouts>
                    <recordTypeMappings>
                       <available>false</available>
                       <defaultRecordTypeMapping>false</defaultRecordTypeMapping>
                       <layoutId>00h30000000doxaAAA</layoutId>
                       <name>0123000000000fzAAA</name>
                       <picklistsForRecordType>
                          <picklistName>Type</picklistName>
                          <picklistValues>
                             <active>true</active>
                             <defaultValue>false</defaultValue>
                             <label>Prospect</label>
                             <value>Prospect</value>
                          </picklistValues>
                          <picklistValues>
                             <active>true</active>
                             <defaultValue>false</defaultValue>
                             <label>Customer - Direct</label>
                             <value>Customer - Direct</value>
                          </picklistValues>
                          <picklistValues>
                             <active>true</active>
                             <defaultValue>false</defaultValue>
                             <label>Customer - Channel</label>
                             <value>Customer - Channel</value>
                          </picklistValues>
                          <picklistValues>
                             <active>true</active>
                             <defaultValue>false</defaultValue>
                             <label>Channel Partner / Reseller</label>
                             <value>Channel Partner / Reseller</value>
                          </picklistValues>
```



```
                        <picklistValues>
                           <active>true</active>
                           <defaultValue>false</defaultValue>
                           <label>Installation Partner</label>
                           <value>Installation Partner</value>
                        </picklistValues>
                        <picklistValues>
                           <active>true</active>
                           <defaultValue>false</defaultValue>
                           <label>Technology Partner</label>
                           <value>Technology Partner</value>
                        </picklistValues>
                        <picklistValues>
                           <active>true</active>
                           <defaultValue>false</defaultValue>
                           <label>Other</label>
                           <value>Other</value>
                        </picklistValues>
                     </picklistsForRecordType>
...
                     <recordTypeId>0123000000000fzAAA</recordTypeId>
                  </recordTypeMappings>
               </result>
            </describeLayoutResponse>
         </soapenv:Body>
      </soapenv:Envelope>
```

Sample SOAP Messages—describeSObject

This topic provides the following sample SOAP messages for the describeSObject call:

- Sample Request Message—describeSObject Call—Enterprise API
- Sample Response Message—describeSObject Call—Enterprise API
- Sample Request Message—describeSObject Call—Partner API
- Sample Response Message—describeSObject Call—Partner API

Sample Request Message—describeSObject Call—Enterprise API

```
POST /services/Soap/c/5.0 HTTP/1.0
Content-Type: text/xml; charset=utf-8
Accept: application/soap+xml, application/dime, multipart/related, text/*
User-Agent: Axis/1.1
Host: na1.salesforce.com
Cache-Control: no-cache
Pragma: no-cache
SOAPAction: ""
Content-Length: 694

<?xml version="1.0" encoding="UTF-8"?>
   <soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
xmlns:xsd="http://www.w3.org/2001/XMLSchema" xmlns:xsi="http://www.w3.org/2001/
XMLSchema-instance">
      <soapenv:Header>
         <ns1:SessionHeader soapenv:mustUnderstand="0"
xmlns:ns1="urn:enterprise.soap.sforce.com">
```

```
            <ns2:sessionId
xmlns:ns2="urn:enterprise.soap.sforce.com">uejMEgecf8yidA3zCjXlNKOSU</
ns2:sessionId>
         </ns1:SessionHeader>
      </soapenv:Header>
      <soapenv:Body>
         <describeSObject xmlns="urn:enterprise.soap.sforce.com">
            <sObjectType>Account</sObjectType>
         </describeSObject>
      </soapenv:Body>
   </soapenv:Envelope>
```

Sample Response Message—describeSObject Call—Enterprise API

```
HTTP/1.0 200 OK Server: sfdc Content-Type: text/xml; charset=utf-8 Date: Wed, 07
Jul 2004 18:11:19 GMT  <?xml version="1.0" encoding="UTF-8"?>
   <soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
xmlns:xsd="http://www.w3.org/2001/XMLSchema" xmlns:xsi="http://www.w3.org/2001/
XMLSchema-instance">
      <soapenv:Body>
         <describeSObjectResponse xmlns="urn:enterprise.soap.sforce.com">
            <result>
               <activateable>false</activateable>
               <createable>true</createable>
               <custom>false</custom>
               <deletable>true</deletable>
               <fields>
                  <autoNumber>false</autoNumber>
                  <byteLength>18</byteLength>
                  <createable>false</createable>
                  <custom>false</custom>
                  <defaultedOnCreate>false</defaultedOnCreate>
                  <digits>0</digits>
                  <filterable>true</filterable>
                  <label>Account ID</label>
                  <length>18</length>
                  <name>Id</name>
                  <nameField>false</nameField>
                  <nillable>false</nillable>
                  <picklistValues xsi:nil="true"/>
                  <precision>0</precision>
                  <referenceTo xsi:nil="true"/>
                  <restrictedPicklist>false</restrictedPicklist>
                  <scale>0</scale>
                  <soapType>tns:ID</soapType>
                  <type>id</type>
                  <updateable>false</updateable>
               </fields>
               <fields>
                  <autoNumber>false</autoNumber>
                  <byteLength>240</byteLength>
                  <createable>true</createable>
                  <custom>false</custom>
                  <defaultedOnCreate>false</defaultedOnCreate>
                  <digits>0</digits>
                  <filterable>true</filterable>
                  <label>Account Name</label>
                  <length>80</length>
                  <name>Name</name>
```



```
      <nameField>true</nameField>
      <nillable>false</nillable>
      <picklistValues xsi:nil="true"/>
      <precision>0</precision>
      <referenceTo xsi:nil="true"/>
      <restrictedPicklist>false</restrictedPicklist>
      <scale>0</scale>
      <soapType>xsd:string</soapType>
      <type>string</type>
      <updateable>true</updateable>
   </fields>
   <fields>
      <autoNumber>false</autoNumber>
      <byteLength>120</byteLength>
      <createable>true</createable>
      <custom>false</custom>
      <defaultedOnCreate>false</defaultedOnCreate>
      <digits>0</digits>
      <filterable>true</filterable>
      <label>Account Type</label>
      <length>40</length>
      <name>Type</name>
      <nameField>false</nameField>
      <nillable>true</nillable>
      <picklistValues>
         <active>true</active>
         <defaultValue>false</defaultValue>
         <label xsi:nil="true"/>
         <value>Prospect</value>
      </picklistValues>
      <picklistValues>
         <active>true</active>
         <defaultValue>false</defaultValue>
         <label xsi:nil="true"/>
         <value>Customer - Direct</value>
      </picklistValues>
      <picklistValues>
         <active>true</active>
         <defaultValue>false</defaultValue>
         <label xsi:nil="true"/>
         <value>Customer - Channel</value>
      </picklistValues>
      <picklistValues>
         <active>true</active>
         <defaultValue>false</defaultValue>
         <label xsi:nil="true"/>
         <value>Channel Partner / Reseller</value>
      </picklistValues>
      <picklistValues>
         <active>true</active>
         <defaultValue>false</defaultValue>
         <label xsi:nil="true"/>
         <value>Installation Partner</value>
      </picklistValues>
      <picklistValues>
         <active>true</active>
         <defaultValue>false</defaultValue>
         <label xsi:nil="true"/>
         <value>Technology Partner</value>
      </picklistValues>
```



```
                  <picklistValues>
                     <active>true</active>
                     <defaultValue>false</defaultValue>
                     <label xsi:nil="true"/>
                     <value>Other</value>
                  </picklistValues>
                  <precision>0</precision>
                  <referenceTo xsi:nil="true"/>
                  <restrictedPicklist>false</restrictedPicklist>
                  <scale>0</scale>
                  <soapType>xsd:string</soapType>
                  <type>picklist</type>
                  <updateable>true</updateable>
               </fields>
               <keyPrefix>001</keyPrefix>
               <label>Account</label>
               <name>Account</name>
               <queryable>true</queryable>
               <replicateable>true</replicateable>
               <retrieveable>true</retrieveable>
               <searchable>true</searchable>
               <undeletable>false</undeletable>
               <updateable>true</updateable>
              <urlDetail>https://blitzna1.eng.salesforce.com/{ID}</urlDetail>
               <urlEdit>https://blitzna1.eng.salesforce.com/{ID}/e</urlEdit>
               <urlNew>https://blitzna1.eng.salesforce.com/001/e</urlNew>
            </result>
         </describeSObjectResponse>
      </soapenv:Body>
   </soapenv:Envelope>
```

Sample Request Message—describeSObject Call—Partner API

```
POST /services/Soap/c/5.0 HTTP/1.0
Content-Type: text/xml; charset=utf-8
Accept: application/soap+xml, application/dime, multipart/related, text/*
User-Agent: Axis/1.1
Host: na1.salesforce.com
Cache-Control: no-cache
Pragma: no-cache
SOAPAction: ""
Content-Length: 694

<?xml version="1.0" encoding="UTF-8"?>
   <soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
xmlns:xsd="http://www.w3.org/2001/XMLSchema" xmlns:xsi="http://www.w3.org/2001/
XMLSchema-instance">
      <soapenv:Header>
       <ns1:SessionHeader soapenv:mustUnderstand="0" xmlns:ns1="SforceService">
            <ns2:sessionId
xmlns:ns2="urn:enterprise.soap.sforce.com">uejMEgecf8yidA3zCjXlNKOSU</
ns2:sessionId>
         </ns1:SessionHeader>
      </soapenv:Header>
      <soapenv:Body>
         <describeSObject xmlns="urn:enterprise.soap.sforce.com">
            <sObjectType>Account</sObjectType>
         </describeSObject>
      </soapenv:Body>
```



```
   </soap:Envelope>
```

Sample Response Message—describeSObject Call—Partner API

```
HTTP/1.0 200 OK Server: sfdc Content-Type: text/xml; charset=utf-8 Date: Wed, 07
Jul 2004 18:11:19 GMT  <?xml version="1.0" encoding="UTF-8"?>
   <soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
xmlns:xsd="http://www.w3.org/2001/XMLSchema" xmlns:xsi="http://www.w3.org/2001/
XMLSchema-instance">
      <soapenv:Body>
         <describeSObjectResponse xmlns="urn:enterprise.soap.sforce.com">
            <result>
               <activateable>false</activateable>
               <createable>true</createable>
               <custom>false</custom>
               <deletable>true</deletable>
               <fields>
                  <autoNumber>false</autoNumber>
                  <byteLength>18</byteLength>
                  <createable>false</createable>
                  <custom>false</custom>
                  <defaultedOnCreate>false</defaultedOnCreate>
                  <digits>0</digits>
                  <filterable>true</filterable>
                  <label>Account ID</label>
                  <length>18</length>
                  <name>Id</name>
                  <nameField>false</nameField>
                  <nillable>false</nillable>
                  <picklistValues xsi:nil="true"/>
                  <precision>0</precision>
                  <referenceTo xsi:nil="true"/>
                  <restrictedPicklist>false</restrictedPicklist>
                  <scale>0</scale>
                  <soapType>tns:ID</soapType>
                  <type>id</type>
                  <updateable>false</updateable>
               </fields>
               <fields>
                  <autoNumber>false</autoNumber>
                  <byteLength>240</byteLength>
                  <createable>true</createable>
                  <custom>false</custom>
                  <defaultedOnCreate>false</defaultedOnCreate>
                  <digits>0</digits>
                  <filterable>true</filterable>
                  <label>Account Name</label>
                  <length>80</length>
                  <name>Name</name>
                  <nameField>true</nameField>
                  <nillable>false</nillable>
                  <picklistValues xsi:nil="true"/>
                  <precision>0</precision>
                  <referenceTo xsi:nil="true"/>
                  <restrictedPicklist>false</restrictedPicklist>
                  <scale>0</scale>
                  <soapType>xsd:string</soapType>
                  <type>string</type>
                  <updateable>true</updateable>
```



```
</fields>
<fields>
   <autoNumber>false</autoNumber>
   <byteLength>120</byteLength>
   <createable>true</createable>
   <custom>false</custom>
   <defaultedOnCreate>false</defaultedOnCreate>
   <digits>0</digits>
   <filterable>true</filterable>
   <label>Account Type</label>
   <length>40</length>
   <name>Type</name>
   <nameField>false</nameField>
   <nillable>true</nillable>
   <picklistValues>
      <active>true</active>
      <defaultValue>false</defaultValue>
      <label xsi:nil="true"/>
      <value>Prospect</value>
   </picklistValues>
   <picklistValues>
      <active>true</active>
      <defaultValue>false</defaultValue>
      <label xsi:nil="true"/>
      <value>Customer - Direct</value>
   </picklistValues>
   <picklistValues>
      <active>true</active>
      <defaultValue>false</defaultValue>
      <label xsi:nil="true"/>
      <value>Customer - Channel</value>
   </picklistValues>
   <picklistValues>
      <active>true</active>
      <defaultValue>false</defaultValue>
      <label xsi:nil="true"/>
      <value>Channel Partner / Reseller</value>
   </picklistValues>
   <picklistValues>
      <active>true</active>
      <defaultValue>false</defaultValue>
      <label xsi:nil="true"/>
      <value>Installation Partner</value>
   </picklistValues>
   <picklistValues>
      <active>true</active>
      <defaultValue>false</defaultValue>
      <label xsi:nil="true"/>
      <value>Technology Partner</value>
   </picklistValues>
   <picklistValues>
      <active>true</active>
      <defaultValue>false</defaultValue>
      <label xsi:nil="true"/>
      <value>Other</value>
   </picklistValues>
   <precision>0</precision>
   <referenceTo xsi:nil="true"/>
   <restrictedPicklist>false</restrictedPicklist>
   <scale>0</scale>
```



```
                  <soapType>xsd:string</soapType>
                  <type>picklist</type>
                  <updateable>true</updateable>
               </fields>
               <keyPrefix>001</keyPrefix>
               <label>Account</label>
               <name>Account</name>
               <queryable>true</queryable>
               <replicateable>true</replicateable>
               <retrieveable>true</retrieveable>
               <searchable>true</searchable>
               <undeletable>false</undeletable>
               <updateable>true</updateable>
              <urlDetail>https://blitzna1.eng.salesforce.com/{ID}</urlDetail>
               <urlEdit>https://blitzna1.eng.salesforce.com/{ID}/e</urlEdit>
               <urlNew>https://blitzna1.eng.salesforce.com/001/e</urlNew>
            </result>
         </describeSObjectResponse>
      </soapenv:Body>
   </soapenv:Envelope>
```

Sample SOAP Messages—getDeleted

This topic provides the following sample SOAP messages for the getDeleted call:

- Sample Request Message—getDeleted Call—Enterprise API
- Sample Response Message—getDeleted Call—Enterprise API
- Sample Request Message—getDeleted Call—Partner API
- Sample Response Message—getDeleted Call—Partner API

Sample Request Message—getDeleted Call—Enterprise API

```
POST /services/Soap/c/5.0 HTTP/1.0
Content-Type: text/xml; charset=utf-8
Accept: application/soap+xml, application/dime, multipart/related, text/*
User-Agent: Axis/1.1
Host: na1.salesforce.com
Cache-Control: no-cache
Pragma: no-cache
SOAPAction: ""
Content-Length: 830

<?xml version="1.0" encoding="UTF-8"?>
   <soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
xmlns:xsd="http://www.w3.org/2001/XMLSchema" xmlns:xsi="http://www.w3.org/2001/
XMLSchema-instance">
      <soapenv:Header>
         <ns1:SessionHeader soapenv:mustUnderstand="0"
xmlns:ns1="urn:enterprise.soap.sforce.com">
            <ns2:sessionId
xmlns:ns2="urn:enterprise.soap.sforce.com">VAHXuWPZ5ODayiABjapEs3ogoK_jXlNKOSU</
ns2:sessionId>
         </ns1:SessionHeader>
      </soapenv:Header>
      <soapenv:Body>
         <getDeleted xmlns="urn:enterprise.soap.sforce.com">
            <sObjectType>Account</sObjectType>
```



```
            <startDate xsi:type="xsd:dateTime">2004-07-07T18:20:01.567Z</
startDate>
            <endDate xsi:type="xsd:dateTime">2004-07-07T18:25:01.567Z</endDate>
         </getDeleted>
      </soapenv:Body>
   </soapenv:Envelope>
```

Sample Response Message—getDeleted Call—Enterprise API

```
HTTP/1.0 200 OK
Server: sfdc
Content-Type: text/xml; charset=utf-8
Date: Wed, 07 Jul 2004 18:25:01 GMT

<?xml version="1.0" encoding="UTF-8"?>
   <soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
xmlns:xsd="http://www.w3.org/2001/XMLSchema" xmlns:xsi="http://www.w3.org/2001/
XMLSchema-instance">
      <soapenv:Body>
         <getDeletedResponse xmlns="urn:enterprise.soap.sforce.com">
            <result>
               <deletedRecords>
                  <deletedDate>2004-07-07T18:24:11.000Z</deletedDate>
                  <id>00130000002K9TvAAK</id>
               </deletedRecords>
               <deletedRecords>
                  <deletedDate>2004-07-07T18:24:11.000Z</deletedDate>
                  <id>00130000002K9TwAAK</id>
               </deletedRecords>
               <deletedRecords>
                  <deletedDate>2004-07-07T18:24:11.000Z</deletedDate>
                  <id>00130000002K9TtAAK</id>
               </deletedRecords>
               <deletedRecords>
                  <deletedDate>2004-07-07T18:24:11.000Z</deletedDate>
                  <id>00130000002K9TuAAK</id>
               </deletedRecords>
            </result>
         </getDeletedResponse>
      </soapenv:Body>
   </soapenv:Envelope>
```

Sample Request Message—getDeleted Call—Partner API

```
POST https://na1.salesforce.com/services/Soap/u/5.0 HTTP/1.1
User-Agent: Mozilla/4.0 (compatible; MSIE 6.0; MS Web Services Client Protocol
1.1.4322.573)
Content-Type: text/xml; charset=utf-8
SOAPAction: ""
Content-Length: 662
Expect: 100-continue
Host: na1.salesforce.com

<?xml version="1.0" encoding="utf-8"?>
   <soap:Envelope xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xmlns:xsd="http://
www.w3.org/2001/XMLSchema">
```



```
      <soap:Header>
         <SessionHeader xmlns="urn:partner.soap.sforce.com">
            <sessionId>UjcSCQ0Qq1EbAcE51vYkEuQr69_Igrhm90_jXlNKOSU</sessionId>
         </SessionHeader>
      </soap:Header>
      <soap:Body>
         <getDeleted xmlns="urn:partner.soap.sforce.com">
            <sObjectType>Account</sObjectType>
            <startDate>2004-07-07T10:05:37.8540000-07:00</startDate>
            <endDate>2004-07-07T10:10:37.8540000-07:00</endDate>
         </getDeleted>
      </soap:Body>
   </soap:Envelope>
```

Sample Response Message—getDeleted Call—Partner API

```
HTTP/1.1 200 OK
Server: sfdc
Content-Type: text/xml; charset=utf-8
Transfer-Encoding: chunked
Date: Wed, 07 Jul 2004 17:10:44 GMT

<?xml version="1.0" encoding="UTF-8"?>
   <soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
xmlns:xsd="http://www.w3.org/2001/XMLSchema" xmlns:xsi="http://www.w3.org/2001/
XMLSchema-instance">
      <soapenv:Body>
         <getDeletedResponse xmlns="urn:partner.soap.sforce.com">
            <result>
               <deletedRecords>
                  <deletedDate>2004-07-07T17:07:20.000Z</deletedDate>
                  <id>00130000002K9RrAAK</id>
               </deletedRecords>
               <deletedRecords>
                  <deletedDate>2004-07-07T17:07:21.000Z</deletedDate>
                  <id>00130000002K9RsAAK</id>
               </deletedRecords>
            </result>
         </getDeletedResponse>
      </soapenv:Body>
   </soapenv:Envelope>
```

Sample SOAP Messages—getServerTimestamp

This topic provides the following sample SOAP messages for the getServerTimestamp call:

- Sample Request Message—getServerTimestamp Call—Enterprise API
- Sample Response Message—getServerTimestamp Call—Enterprise API
- Sample Request Message—getServerTimestamp Call—Partner API
- Sample Response Message—getServerTimestamp Call—Partner API

Sample Request Message—getServerTimestamp Call—Enterprise API

```
POST https://na1.salesforce.com/services/Soap/c/5.0 HTTP/1.1
User-Agent: Mozilla/4.0 (compatible; MSIE 6.0; MS Web Services Client Protocol
1.1.4322.573)
```



```
Content-Type: text/xml; charset=utf-8
SOAPAction: ""
Content-Length: 523
Expect: 100-continue
Host: na1.salesforce.com

   <?xml version="1.0" encoding="utf-8"?>
      <soap:Envelope xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xmlns:xsd="http://
www.w3.org/2001/XMLSchema">
         <soap:Header>
            <SessionHeader xmlns="urn:enterprise.soap.sforce.com">
               <sessionId>z1E_NI80RXGyM48GB2etyLuR8DBelyRk_MTjFeOxPed0_jXlNKOSU</
sessionId>
            </SessionHeader>
         </soap:Header>
         <soap:Body>
            <getServerTimestamp xmlns="urn:enterprise.soap.sforce.com" />
         </soap:Body>
      </soap:Envelope>
```

Sample Response Message—getServerTimestamp Call—Enterprise API

```
HTTP/1.1 200 OK
Server: sfdc
Content-Type: text/xml; charset=utf-8
Transfer-Encoding: chunked
Date: Wed, 07 Jul 2004 17:36:19 GMT

   <?xml version="1.0" encoding="UTF-8"?>
      <soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
xmlns:xsd="http://www.w3.org/2001/XMLSchema" xmlns:xsi="http://www.w3.org/2001/
XMLSchema-instance">
         <soapenv:Body>
            <getServerTimestampResponse xmlns="urn:enterprise.soap.sforce.com">
               <result>
                  <timestamp>2004-07-07T17:36:19.825Z</timestamp>
               </result>
            </getServerTimestampResponse>
         </soapenv:Body>
      </soapenv:Envelope>
```

Sample Request Message—getServerTimestamp Call—Partner API

```
POST https://na1.salesforce.com/services/Soap/u/5.0 HTTP/1.1
User-Agent: Mozilla/4.0 (compatible; MSIE 6.0; MS Web Services Client Protocol
1.1.4322.573)
Content-Type: text/xml; charset=utf-8
SOAPAction: ""
Content-Length: 517
Expect: 100-continue
Host: na1.salesforce.com

<?xml version="1.0" encoding="utf-8"?>
   <soap:Envelope xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xmlns:xsd="http://
www.w3.org/2001/XMLSchema">
```



```
      <soap:Header>
         <SessionHeader xmlns="urn:partner.soap.sforce.com">
            <sessionId>MkogrC4M2WUuwaKjLUlPmLaKyTTif.OBZ7vO__R5L</sessionId>
         </SessionHeader>
      </soap:Header>
      <soap:Body>
         <getServerTimestamp xmlns="urn:partner.soap.sforce.com" />
      </soap:Body>
   </soap:Envelope>
```

Sample Response Message—getServerTimestamp Call—Partner API

```
HTTP/1.1 200 OK
Server: sfdc
Content-Type: text/xml; charset=utf-8
Transfer-Encoding: chunked
Date: Wed, 07 Jul 2004 16:26:02 GMT

<?xml version="1.0" encoding="UTF-8"?>
   <soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
xmlns:xsd="http://www.w3.org/2001/XMLSchema" xmlns:xsi="http://www.w3.org/2001/
XMLSchema-instance">
      <soapenv:Body>
         <getServerTimestampResponse xmlns="urn:partner.soap.sforce.com">
            <result>
               <timestamp>2004-07-07T16:26:02.963Z</timestamp>
            </result>
         </getServerTimestampResponse>
      </soapenv:Body>
   </soapenv:Envelope>
```

Sample SOAP Messages—getUpdated

This topic provides the following sample SOAP messages for the getUpdated call:

- Sample Request Message—getUpdated Call—Enterprise API
- Sample Response Message—getUpdated Call—Enterprise API
- Sample Request Message—getUpdated Call—Partner API
- Sample Response Message—getUpdated Call—Partner API

Sample Request Message—getUpdated Call—Enterprise API

```
POST /services/Soap/c/5.0 HTTP/1.0
Content-Type: text/xml; charset=utf-8
Accept: application/soap+xml, application/dime, multipart/related, text/*
User-Agent: Axis/1.1
Host: na1.salesforce.com
Cache-Control: no-cache
Pragma: no-cache
SOAPAction: ""
Content-Length: 830

<?xml version="1.0" encoding="UTF-8"?>
   <soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
xmlns:xsd="http://www.w3.org/2001/XMLSchema" xmlns:xsi="http://www.w3.org/2001/
XMLSchema-instance">
```



```
      <soapenv:Header>
         <ns1:SessionHeader soapenv:mustUnderstand="0"
xmlns:ns1="urn:enterprise.soap.sforce.com">
            <ns2:sessionId
xmlns:ns2="urn:enterprise.soap.sforce.com">VAHXuWPZ5ODayiABjapEs3og_jXlNKOSU</
ns2:sessionId>
         </ns1:SessionHeader>
      </soapenv:Header>
      <soapenv:Body>
         <getUpdated xmlns="urn:enterprise.soap.sforce.com">
            <sObjectType>Account</sObjectType>
            <startDate xsi:type="xsd:dateTime">2004-07-07T18:18:03.604Z</
startDate>
            <endDate xsi:type="xsd:dateTime">2004-07-07T18:23:03.604Z</endDate>
         </getUpdated>
      </soapenv:Body>
   </soapenv:Envelope>
```

Sample Response Message—getUpdated Call—Enterprise API

```
HTTP/1.0 200 OK
Server: sfdc
Content-Type: text/xml; charset=utf-8
Date: Wed, 07 Jul 2004 18:23:04 GMT

<?xml version="1.0" encoding="UTF-8"?>
   <soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
xmlns:xsd="http://www.w3.org/2001/XMLSchema" xmlns:xsi="http://www.w3.org/2001/
XMLSchema-instance">
      <soapenv:Body>
         <getUpdatedResponse xmlns="urn:enterprise.soap.sforce.com">
            <result>
               <ids>00130000002K9TtAAK</ids>
               <ids>00130000002K9TuAAK</ids>
               <ids>00130000002K9TvAAK</ids>
               <ids>00130000002K9TwAAK</ids>
            </result>
         </getUpdatedResponse>
      </soapenv:Body>
   </soapenv:Envelope>
```

Sample Request Message—getUpdated Call—Partner API

```
POST https://na1.salesforce.com/services/Soap/u/5.0 HTTP/1.1
User-Agent: Mozilla/4.0 (compatible; MSIE 6.0; MS Web Services Client Protocol
1.1.4322.573)
Content-Type: text/xml; charset=utf-8
SOAPAction: ""
Content-Length: 662
Expect: 100-continue
Host: na1.salesforce.com

<?xml version="1.0" encoding="utf-8"?>
   <soap:Envelope xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xmlns:xsd="http://
www.w3.org/2001/XMLSchema">
      <soap:Header>
```

```
         <SessionHeader xmlns="urn:partner.soap.sforce.com">
            <sessionId>4I1yNabX_5FyOGOnuFO64VpRP079_jXlNKOSU</sessionId>
         </SessionHeader>
      </soap:Header>
      <soap:Body>
         <getUpdated xmlns="urn:partner.soap.sforce.com">
            <sObjectType>Account</sObjectType>
            <startDate>2004-07-07T10:00:41.9650000-07:00</startDate>
            <endDate>2004-07-07T10:05:41.9650000-07:00</endDate>
         </getUpdated>
      </soap:Body>
   </soap:Envelope>
```

Sample Response Message—getUpdated Call—Partner API

```
HTTP/1.1 200 OK
Server: sfdc
Content-Type: text/xml; charset=utf-8
Transfer-Encoding: chunked
Date: Wed, 07 Jul 2004 17:05:48 GMT

<?xml version="1.0" encoding="UTF-8"?>
   <soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
xmlns:xsd="http://www.w3.org/2001/XMLSchema" xmlns:xsi="http://www.w3.org/2001/
XMLSchema-instance">
      <soapenv:Body>
         <getUpdatedResponse xmlns="urn:partner.soap.sforce.com">
            <result>
               <ids>00130000000ZdoSAAS</ids>
               <ids>00130000002K9RrAAK</ids>
               <ids>00130000002K9RsAAK</ids>
            </result>
         </getUpdatedResponse>
      </soapenv:Body>
   </soapenv:Envelope>
```

Sample SOAP Messages—getUserInfo

This topic provides the following sample SOAP messages for the getUserInfo call:

- Sample Request Message—getUserInfo Call—Enterprise API
- Sample Response Message—getUserInfo Call—Enterprise API
- Sample Request Message—getUserInfo Call—Partner API
- Sample Response Message—getUserInfo Call—Partner API

Sample Request Message—getUserInfo Call—Enterprise API

```
POST https://na1.salesforce.com/services/Soap/c/5.0 HTTP/1.1
User-Agent: Mozilla/4.0 (compatible; MSIE 6.0; MS Web Services Client Protocol
1.1.4322.573)
Content-Type: text/xml; charset=utf-8
SOAPAction: ""
Content-Length: 516
Expect: 100-continue
Host: na1.salesforce.com
```



```
      <?xml version="1.0" encoding="utf-8"?>
         <soap:Envelope xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xmlns:xsd="http://
www.w3.org/2001/XMLSchema">
            <soap:Header>
               <SessionHeader xmlns="urn:enterprise.soap.sforce.com">

<sessionId>z1E_NI80RXGyM48GB2etyLuR8DBelyRk_MTjFeOxPed0_jXlNKOSU</sessionId>
               </SessionHeader>
            </soap:Header>
            <soap:Body>
               <getUserInfo xmlns="urn:enterprise.soap.sforce.com" />
            </soap:Body>
         </soap:Envelope>
```

Sample Response Message—getUserInfo Call—Enterprise API

```
HTTP/1.1 200 OK
Server: sfdc
Content-Type: text/xml; charset=utf-8
Transfer-Encoding: chunked
Date: Wed, 07 Jul 2004 17:36:19 GMT

      <?xml version="1.0" encoding="UTF-8"?>
         <soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/
envelope/" xmlns:xsd="http://www.w3.org/2001/XMLSchema" xmlns:xsi="http://
www.w3.org/2001/XMLSchema-instance">
            <soapenv:Body>
               <getUserInfoResponse xmlns="urn:enterprise.soap.sforce.com">
                  <result>
                     <currencySymbol xsi:nil="true"/>
                     <organizationId>00D3000000044rV</organizationId>
                   <organizationMultiCurrency>true</organizationMultiCurrency>
                     <organizationName>acme.com</organizationName>
                  <userDefaultCurrencyIsoCode>USD</userDefaultCurrencyIsoCode>
                     <userEmail>user@acme.com</userEmail>
                     <userFullName>Joe Doe</userFullName>
                     <userId>00530000000tt3MAAY</userId>
                     <userLanguage>en_US</userLanguage>
                     <userLocale>en_US</userLocale>
                     <userTimeZone>America/Los_Angeles</userTimeZone>
                  </result>
               </getUserInfoResponse>
            </soapenv:Body>
         </soapenv:Envelope>
```

Sample Request Message—getUserInfo Call—Partner API

```
POST https://na1.salesforce.com/services/Soap/u/5.0 HTTP/1.1
User-Agent: Mozilla/4.0 (compatible; MSIE 6.0; MS Web Services Client Protocol
1.1.4322.573)
Content-Type: text/xml; charset=utf-8
SOAPAction: ""
Content-Length: 510
Expect: 100-continue
Host: na1.salesforce.com
```



```
   <?xml version="1.0" encoding="utf-8"?>
      <soap:Envelope xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xmlns:xsd="http://
www.w3.org/2001/XMLSchema">
         <soap:Header>
            <SessionHeader xmlns="urn:partner.soap.sforce.com">
               <sessionId>MkogrC4M2WUuwaKjLUlPmLaKyTTif.OBZ7vO__R</sessionId>
            </SessionHeader>
         </soap:Header>
         <soap:Body>
            <getUserInfo xmlns="urn:partner.soap.sforce.com" />
         </soap:Body>
      </soap:Envelope>
```

Sample Response Message—getUserInfo Call—Partner API

```
HTTP/1.1 200 OK
Server: sfdc
Content-Type: text/xml; charset=utf-8
Transfer-Encoding: chunked
Date: Wed, 07 Jul 2004 16:26:02 GMT

   <?xml version="1.0" encoding="UTF-8"?>
      <soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
xmlns:xsd="http://www.w3.org/2001/XMLSchema" xmlns:xsi="http://www.w3.org/2001/
XMLSchema-instance">
         <soapenv:Body>
            <getUserInfoResponse xmlns="urn:partner.soap.sforce.com">
               <result>
                  <currencySymbol xsi:nil="true"/>
                  <organizationId>00D3000456008vV</organizationId>
                  <organizationMultiCurrency>true</organizationMultiCurrency>
                  <organizationName>acme.com</organizationName>
                 <userDefaultCurrencyIsoCode>USD</userDefaultCurrencyIsoCode>
                  <userEmail>user@acme.com</userEmail>
                  <userFullName>Joe Doe</userFullName>
                  <userId>00530000000kk5MAAY</userId>
                  <userLanguage>en_US</userLanguage>
                  <userLocale>en_US</userLocale>
                  <userTimeZone>America/Los_Angeles</userTimeZone>
               </result>
            </getUserInfoResponse>
         </soapenv:Body>
      </soapenv:Envelope>
```

Sample SOAP Messages—login

This topic provides the following sample SOAP messages for the login call:

- Sample Request Message—login Call—Enterprise API
- Sample Response Message—login Call—Enterprise API
- Sample Request Message—login Call—Partner API
- Sample Response Message—login Call—Partner API

Sample Request Message—login Call—Enterprise API

```
POST https://www.salesforce.com/services/Soap/c/5.0 HTTP/1.1
User-Agent: Mozilla/4.0 (compatible; MSIE 6.0; MS Web Services Client Protocol
1.1.4322.573)
Content-Type: text/xml; charset=utf-8
SOAPAction: ""
Content-Length: 367
Expect: 100-continue
Host: www.salesforce.com

<?xml version="1.0" encoding="utf-8"?>
   <soap:Envelope xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xmlns:xsd="http://
www.w3.org/2001/XMLSchema">
      <soap:Body>
         <login xmlns="urn:enterprise.soap.sforce.com">
            <username>user@domain.com</username>
            <password>secret</password>
         </login>
      </soap:Body>
   </soap:Envelope>
```

Sample Response Message—login Call—Enterprise API

```
HTTP/1.1 200 OK
Server: sfdc
Content-Type: text/xml; charset=utf-8
Transfer-Encoding: chunked
Date: Wed, 07 Jul 2004 17:36:19 GMT

<?xml version="1.0" encoding="UTF-8"?>
<soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
xmlns:xsd="http://www.w3.org/2001/XMLSchema" xmlns:xsi="http://www.w3.org/2001/
XMLSchema-instance">
 <soapenv:Body>
  <loginResponse xmlns="urn:enterprise.soap.sforce.com">
   <result>
    <passwordExpired>false</passwordExpired>
    <serverUrl>https://blitzna1.eng.salesforce.com/services/Soap/c/5.0</
serverUrl>
    <sessionId>D.GRN2nNCBqwYQEnIO8t0_jXlNKOSU</sessionId>
    <userId>00530000000cJzKAAU</userId>
   </result>
  </loginResponse>
 </soapenv:Body>
</soapenv:Envelope>
```

Sample Request Message—login Call—Partner API

```
POST https://www.salesforce.com/services/Soap/u/5.0 HTTP/1.1
User-Agent: Mozilla/4.0 (compatible; MSIE 6.0; MS Web Services Client Protocol
1.1.4322.573)
Content-Type: text/xml; charset=utf-8
SOAPAction: ""Content-Length: 364
Expect: 100-continue
Proxy-Connection: Keep-Alive
```



```
Host: www.salesforce.com

<?xml version="1.0" encoding="utf-8"?>
   <soap:Envelope xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xmlns:xsd="http://
www.w3.org/2001/XMLSchema">
      <soap:Body>
         <login xmlns="urn:partner.soap.sforce.com">
            <username>user@domain.com</username>
            <password>secret</password>
         </login>
      </soap:Body>
   </soap:Envelope>
```

Sample Response Message—login Call—Partner API

```
HTTP/1.1 200 OK
Server: Resin/3.0.s040331
Content-Type: text/xml; charset=utf-8
Connection: close
Transfer-Encoding: chunked
Date: Wed, 07 Jul 2004 16:26:26 GMT

<?xml version="1.0" encoding="UTF-8"?>
<soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
xmlns:xsd="http://www.w3.org/2001/XMLSchema" xmlns:xsi="http://www.w3.org/2001/
XMLSchema-instance">
 <soapenv:Body>
  <loginResponse xmlns="urn:partner.soap.sforce.com">
   <result>
    <passwordExpired>false</passwordExpired>
    <serverUrl>https://blitzna1.eng.salesforce.com/services/Soap/u/5.0</
serverUrl>
    <sessionId>c0l4g2lI.slq0uexTI3HIIuTTc9.wnYd0_jXlNKOSU</sessionId>
    <userId>00530000000cJzKAAU</userId>
   </result>
  </loginResponse>
 </soapenv:Body>
</soapenv:Envelope>
```

Sample SOAP Messages—query

This topic provides the following sample SOAP messages for the query call:

- Sample Request Message—query Call—Enterprise API
- Sample Response Message—query Call—Enterprise API
- Sample Request Message—query Call—Partner API
- Sample Response Message—query Call—Partner API

Sample Request Message—query Call—Enterprise API

```
POST /services/Soap/c/5.0 HTTP/1.0
Content-Type: text/xml; charset=utf-8
Accept: application/soap+xml, application/dime, multipart/related, text/*
User-Agent: Axis/1.1
Host: na1.salesforce.com
```

```
Cache-Control: no-cache
Pragma: no-cache
SOAPAction: ""
Content-Length: 879

<?xml version="1.0" encoding="UTF-8"?>
   <soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
xmlns:xsd="http://www.w3.org/2001/XMLSchema" xmlns:xsi="http://www.w3.org/2001/
XMLSchema-instance">
      <soapenv:Header>
         <ns1:SessionHeader soapenv:mustUnderstand="0"
xmlns:ns1="urn:enterprise.soap.sforce.com">
            <ns2:sessionId
xmlns:ns2="urn:enterprise.soap.sforce.com">WgMUbbkPmBoH_dI9Z89Nvxm6ojXlNKOSU</
ns2:sessionId>
         </ns1:SessionHeader>
         <ns3:QueryOptions soapenv:mustUnderstand="0" xmlns:ns3="SoapService">
            <ns4:batchSize xmlns:ns4="urn:enterprise.soap.sforce.com">3</
ns4:batchSize>
         </ns3:QueryOptions>
      </soapenv:Header>
      <soapenv:Body>
         <query xmlns="urn:enterprise.soap.sforce.com">
            <queryString>select FirstName, LastName from Contact</queryString>
         </query>
      </soapenv:Body>
   </soapenv:Envelope>
```

Sample Response Message—query Call—Enterprise API

```
HTTP/1.0 200 OK
Server: sfdc
Content-Type: text/xml; charset=utf-8
Date: Wed, 07 Jul 2004 18:04:04 GMT

<?xml version="1.0" encoding="UTF-8"?>
   <soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
xmlns:xsd="http://www.w3.org/2001/XMLSchema" xmlns:xsi="http://www.w3.org/2001/
XMLSchema-instance">
      <soapenv:Body>
         <queryResponse xmlns="urn:enterprise.soap.sforce.com">
            <result>
               <done>false</done>
               <queryLocator>01g30000000590JAAQ-3</queryLocator>
               <records xsi:type="sf:Contact"
xmlns:sf="urn:sobject.enterprise.soap.sforce.com">
                  <sf:FirstName>Merce</sf:FirstName>
                  <sf:LastName>Carroll</sf:LastName>
               </records>
               <records xsi:type="sf:Contact"
xmlns:sf="urn:sobject.enterprise.soap.sforce.com">
                  <sf:FirstName>Jim</sf:FirstName>
                  <sf:LastName>Jones</sf:LastName>
               </records>
               <records xsi:type="sf:Contact"
xmlns:sf="urn:sobject.enterprise.soap.sforce.com">
                  <sf:FirstName>Sydney</sf:FirstName>
                  <sf:LastName>Carroll</sf:LastName>
               </records>
```



```
            <size>82</size>
         </result>
      </queryResponse>
   </soapenv:Body>
</soapenv:Envelope>
```

Sample Request Message—query Call—Partner API

```
POST https://na1.salesforce.com/services/Soap/u/5.0 HTTP/1.1
User-Agent: Mozilla/4.0 (compatible; MSIE 6.0; MS Web Services Client Protocol
1.1.4322.573)
Content-Type: text/xml; charset=utf-8
SOAPAction: ""
Content-Length: 691
Expect: 100-continue
Host: na1.salesforce.com

<?xml version="1.0" encoding="utf-8"?>
   <soap:Envelope xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xmlns:xsd="http://
www.w3.org/2001/XMLSchema">
      <soap:Header>
         <QueryOptions xmlns="urn:partner.soap.sforce.com">
            <batchSize>3</batchSize>
         </QueryOptions>
         <SessionHeader xmlns="urn:partner.soap.sforce.com">
            <sessionId>4I1yNabX_5FyOGOnuFO64VpRP0791eQNkq3V_jXlNKOSU</sessionId>
         </SessionHeader>
      </soap:Header>
      <soap:Body>
         <query xmlns="urn:partner.soap.sforce.com">
           <queryString>select id, Website, Name from Account where Name = 'Golden
Straw'</queryString>
         </query>
      </soap:Body>
   </soap:Envelope>
```

Sample Response Message—query Call—Partner API

```
HTTP/1.1 200 OK
Server: sfdc
Content-Type: text/xml; charset=utf-8
Transfer-Encoding: chunked
Date: Wed, 07 Jul 2004 16:42:42 GMT

<?xml version="1.0" encoding="UTF-8"?>
   <soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
xmlns:xsd="http://www.w3.org/2001/XMLSchema" xmlns:xsi="http://www.w3.org/2001/
XMLSchema-instance">
      <soapenv:Body>
         <queryResponse xmlns="urn:partner.soap.sforce.com">
            <result>
               <done>false</done>
               <queryLocator>01g30000000590GAAQ-3</queryLocator>
               <records xsi:type="sf:sObject"
xmlns:sf="urn:sobject.partner.soap.sforce.com">
                  <sf:type>Account</sf:type>
```

```
                  <sf:Id>001300000019kt7AAA</sf:Id>
                  <sf:Id>001300000019kt7AAA</sf:Id>
                  <sf:Website>www.oz.com</sf:Website>
                  <sf:Name>Golden Straw</sf:Name>
               </records>
               <records xsi:type="sf:sObject"
xmlns:sf="urn:sobject.partner.soap.sforce.com">
                  <sf:type>Account</sf:type>
                  <sf:Id>00130000001rw25AAA</sf:Id>
                  <sf:Id>00130000001rw25AAA</sf:Id>
                  <sf:Website>www.oz.com</sf:Website>
                  <sf:Name>Golden Straw</sf:Name>
               </records>
               <records xsi:type="sf:sObject"
xmlns:sf="urn:sobject.partner.soap.sforce.com">
                  <sf:type>Account</sf:type>
                  <sf:Id>00130000001rwnbAAA</sf:Id>
                  <sf:Id>00130000001rwnbAAA</sf:Id>
                  <sf:Website>www.oz.com</sf:Website>
                  <sf:Name>Golden Straw</sf:Name>
               </records>
               <size>19</size>
            </result>
         </queryResponse>
      </soapenv:Body>
   </soapenv:Envelope>
```

Sample SOAP Messages—queryMore

This topic provides the following sample SOAP messages for the queryMore call:

- Sample Request Message—queryMore Call—Enterprise API
- Sample Response Message—queryMore Call—Enterprise API
- Sample Request Message—queryMore Call—Partner API
- Sample Response Message—queryMore Call—Partner API

Sample Request Message—queryMore Call—Enterprise API

```
POST /services/Soap/c/5.0 HTTP/1.0
Content-Type: text/xml; charset=utf-8
Accept: application/soap+xml, application/dime, multipart/related, text/*
User-Agent: Axis/1.1
Host: na1.salesforce.com
Cache-Control: no-cache
Pragma: no-cache
SOAPAction: ""
Content-Length: 870

<?xml version="1.0" encoding="UTF-8"?>
   <soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
xmlns:xsd="http://www.w3.org/2001/XMLSchema" xmlns:xsi="http://www.w3.org/2001/
XMLSchema-instance">
      <soapenv:Header>
         <ns1:SessionHeader soapenv:mustUnderstand="0"
xmlns:ns1="urn:enterprise.soap.sforce.com">
```

```
            <ns2:sessionId
xmlns:ns2="urn:enterprise.soap.sforce.com">WgMUbbkPmBoH_dI9Z89Nvxm_jXlNKOSU</
ns2:sessionId>
         </ns1:SessionHeader>
         <ns3:QueryOptions soapenv:mustUnderstand="0" xmlns:ns3="SoapService">
            <ns4:batchSize xmlns:ns4="urn:enterprise.soap.sforce.com">3</
ns4:batchSize>
         </ns3:QueryOptions>
      </soapenv:Header>
      <soapenv:Body>
         <queryMore xmlns="urn:enterprise.soap.sforce.com">
            <queryLocator>01g30000000590JAAQ-3</queryLocator>
         </queryMore>
      </soapenv:Body>
   </soapenv:Envelope>
```

Sample Response Message—queryMore Call—Enterprise API

```
HTTP/1.0 200 OK
Server: sfdc
Content-Type: text/xml; charset=utf-8
Date: Wed, 07 Jul 2004 18:04:04 GMT

<?xml version="1.0" encoding="UTF-8"?>
   <soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
xmlns:xsd="http://www.w3.org/2001/XMLSchema" xmlns:xsi="http://www.w3.org/2001/
XMLSchema-instance">
      <soapenv:Body>
         <queryMoreResponse xmlns="urn:enterprise.soap.sforce.com">
            <result>
               <done>false</done>
               <queryLocator>01g30000000590JAAQ-6</queryLocator>
               <records xsi:type="sf:Contact"
xmlns:sf="urn:sobject.enterprise.soap.sforce.com">
                  <sf:FirstName>Query</sf:FirstName>
                  <sf:LastName>Nested04</sf:LastName>
               </records>
               <records xsi:type="sf:Contact"
xmlns:sf="urn:sobject.enterprise.soap.sforce.com">
                  <sf:FirstName>Query</sf:FirstName>
                  <sf:LastName>Nested10</sf:LastName>
               </records>
               <records xsi:type="sf:Contact"
xmlns:sf="urn:sobject.enterprise.soap.sforce.com">
                  <sf:FirstName>Query</sf:FirstName>
                  <sf:LastName>Nested22</sf:LastName>
               </records>
               <size>82</size>
            </result>
         </queryMoreResponse>
      </soapenv:Body>
   </soapenv:Envelope>
```

Sample Request Message—queryMore Call—Partner API

```
POST https://na1.salesforce.com/services/Soap/u/5.0 HTTP/1.1
```



```
User-Agent: Mozilla/4.0 (compatible; MSIE 6.0; MS Web Services Client Protocol
1.1.4322.573)
Content-Type: text/xml; charset=utf-8
SOAPAction: ""
Content-Length: 656
Expect: 100-continue
Host: na1.salesforce.com

      <?xml version="1.0" encoding="utf-8"?>
         <soap:Envelope xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xmlns:xsd="http://
www.w3.org/2001/XMLSchema">
            <soap:Header>
               <QueryOptions xmlns="urn:partner.soap.sforce.com">
                  <batchSize>3</batchSize>
               </QueryOptions>
               <SessionHeader xmlns="urn:partner.soap.sforce.com">
                  <sessionId>4I1yNabX_5FyOGOnuFO64VpRP079_jXlNKOSU</sessionId>
               </SessionHeader>
            </soap:Header>
            <soap:Body>
               <queryMore xmlns="urn:partner.soap.sforce.com">
                  <queryLocator>01g30000000590HAAQ-3</queryLocator>
               </queryMore>
            </soap:Body>
         </soap:Envelope>
```

Sample Response Message—queryMore Call—Partner API

```
HTTP/1.1 200 OK
Server: sfdc
Content-Type: text/xml; charset=utf-8
Transfer-Encoding: chunked
Date: Wed, 07 Jul 2004 16:42:42 GMT

      <?xml version="1.0" encoding="UTF-8"?>
         <soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/
envelope/" xmlns:xsd="http://www.w3.org/2001/XMLSchema" xmlns:xsi="http://
www.w3.org/2001/XMLSchema-instance">
            <soapenv:Body>
               <queryMoreResponse xmlns="urn:partner.soap.sforce.com">
                  <result>
                     <done>false</done>
                     <queryLocator>01g30000000590HAAQ-6</queryLocator>
                     <records xsi:type="sf:sObject"
xmlns:sf="urn:sobject.partner.soap.sforce.com">
                        <sf:type>Contact</sf:type>
                        <sf:FirstName>Query</sf:FirstName>
                        <sf:LastName>Nested04</sf:LastName>
                     </records>
                     <records xsi:type="sf:sObject"
xmlns:sf="urn:sobject.partner.soap.sforce.com">
                        <sf:type>Contact</sf:type>
                        <sf:FirstName>Query</sf:FirstName>
                        <sf:LastName>Nested10</sf:LastName>
                     </records>
                     <records xsi:type="sf:sObject"
xmlns:sf="urn:sobject.partner.soap.sforce.com">
                        <sf:type>Contact</sf:type>
```



```
                        <sf:FirstName>Query</sf:FirstName>
                        <sf:LastName>Nested22</sf:LastName>
                     </records>
                     <size>80</size>
                  </result>
               </queryMoreResponse>
            </soapenv:Body>
         </soapenv:Envelope>
```

Sample SOAP Messages—resetPassword

This topic provides the following sample SOAP messages for the resetPassword call:

- Sample Request Message—resetPassword Call—Enterprise API
- Sample Response Message—resetPassword Call—Enterprise API
- Sample Request Message—resetPassword Call—Partner API
- Sample Response Message—resetPassword Call—Partner API

Sample Request Message—resetPassword Call—Enterprise API

```
POST /services/Soap/c/5.0 HTTP/1.0
Content-Type: text/xml; charset=utf-8
Accept: application/soap+xml, application/dime, multipart/related, text/*
User-Agent: Axis/1.1
Host: na1.salesforce.com
Cache-Control: no-cache
Pragma: no-cache
SOAPAction: ""
Content-Length: 691

<?xml version="1.0" encoding="UTF-8"?>
   <soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
xmlns:xsd="http://www.w3.org/2001/XMLSchema" xmlns:xsi="http://www.w3.org/2001/
XMLSchema-instance">
      <soapenv:Header>
         <ns1:SessionHeader soapenv:mustUnderstand="0"
xmlns:ns1="urn:enterprise.soap.sforce.com">
            <ns2:sessionId
xmlns:ns2="urn:enterprise.soap.sforce.com">uejMEgejXlNKOSU</ns2:sessionId>
         </ns1:SessionHeader>
      </soapenv:Header>
      <soapenv:Body>
         <resetPassword xmlns="urn:enterprise.soap.sforce.com">
            <userId>00530000000e4rO3AAI</userId>
         </resetPassword>
      </soapenv:Body>
   </soapenv:Envelope>
```

Sample Response Message—resetPassword Call—Enterprise API

```
HTTP/1.0 200 OK
Server: sfdc
Content-Type: text/xml; charset=utf-8
Date: Wed, 07 Jul 2004 18:17:20 GMT

<?xml version="1.0" encoding="UTF-8"?>
```



```
   <soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
xmlns:xsd="http://www.w3.org/2001/XMLSchema" xmlns:xsi="http://www.w3.org/2001/
XMLSchema-instance">
      <soapenv:Body>
         <resetPasswordResponse xmlns="urn:enterprise.soap.sforce.com">
            <result>
               <password>g2KvS</password>
            </result>
         </resetPasswordResponse>
      </soapenv:Body>
   </soapenv:Envelope>
```

Sample Request Message—resetPassword Call—Partner API

```
POST https://na1.salesforce.com/services/Soap/u/5.0 HTTP/1.1
User-Agent: Mozilla/4.0 (compatible; MSIE 6.0; MS Web Services Client Protocol
1.1.4322.573)
Content-Type: text/xml; charset=utf-8
SOAPAction: ""
Content-Length: 561
Expect: 100-continue
Host: na1.salesforce.com

         <?xml version="1.0" encoding="utf-8"?>
            <soap:Envelope xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xmlns:xsd="http://
www.w3.org/2001/XMLSchema">
               <soap:Header>
                  <SessionHeader xmlns="urn:partner.soap.sforce.com">
                     <sessionId>4I1yNabX_5FyOGOn_jXlNKOSU</sessionId>
                  </SessionHeader>
               </soap:Header>
               <soap:Body>
                  <resetPassword xmlns="urn:partner.soap.sforce.com">
                     <userId>0053000000088f3AAI</userId>
                  </resetPassword>
               </soap:Body>
            </soap:Envelope>
```

Sample Response Message—resetPassword Call—Partner API

```
HTTP/1.1 200 OK
Server: sfdc
Content-Type: text/xml; charset=utf-8
Transfer-Encoding: chunked
Date: Wed, 07 Jul 2004 16:57:57 GMT

         <?xml version="1.0" encoding="UTF-8"?>
            <soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/
envelope/" xmlns:xsd="http://www.w3.org/2001/XMLSchema" xmlns:xsi="http://
www.w3.org/2001/XMLSchema-instance">
               <soapenv:Body>
                  <resetPasswordResponse xmlns="urn:partner.soap.sforce.com">
                     <result>
                        <password>AKxJc</password>
                     </result>
                  </resetPasswordResponse>
```



```
            </soapenv:Body>
         </soapenv:Envelope>
```

Sample SOAP Messages—retrieve

This topic provides the following sample SOAP messages for the retrieve call:

- Sample Request Message—retrieve Call—Enterprise API
- Sample Response Message—retrieve Call—Enterprise API
- Sample Request Message—retrieve Call—Partner API
- Sample Response Message—retrieve Call—Partner API

Sample Request Message—retrieve Call—Enterprise API

```
POST /services/Soap/c/5.0 HTTP/1.0
Content-Type: text/xml; charset=utf-8
Accept: application/soap+xml, application/dime, multipart/related, text/*
User-Agent: Axis/1.1
Host: na1.salesforce.com
Cache-Control: no-cache
Pragma: no-cache
SOAPAction: ""
Content-Length: 805

<?xml version="1.0" encoding="UTF-8"?>
   <soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
xmlns:xsd="http://www.w3.org/2001/XMLSchema" xmlns:xsi="http://www.w3.org/2001/
XMLSchema-instance">
      <soapenv:Header>
         <ns1:SessionHeader soapenv:mustUnderstand="0"
xmlns:ns1="urn:enterprise.soap.sforce.com">
            <ns2:sessionId
xmlns:ns2="urn:enterprise.soap.sforce.com">uejMEgecf8yidA3zCFeT6baks1jXlNKOSU</
ns2:sessionId>
         </ns1:SessionHeader>
      </soapenv:Header>
      <soapenv:Body>
         <retrieve xmlns="urn:enterprise.soap.sforce.com">
            <fieldList>Id, AccountNumber, Name, Website</fieldList>
            <sObjectType>Account</sObjectType>
            <ids>00130000002K9TrAAK</ids>
            <ids>00130000002K9TsAAK</ids>
         </retrieve>
      </soapenv:Body>
   </soapenv:Envelope>
```

Sample Response Message—retrieve Call—Enterprise API

```
HTTP/1.0 200 OK
Server: sfdc
Content-Type: text/xml; charset=utf-8
Date: Wed, 07 Jul 2004 18:15:59 GMT

<?xml version="1.0" encoding="UTF-8"?>
```

```
   <soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
xmlns:xsd="http://www.w3.org/2001/XMLSchema" xmlns:xsi="http://www.w3.org/2001/
XMLSchema-instance">
      <soapenv:Body>
         <retrieveResponse xmlns="urn:enterprise.soap.sforce.com">
            <result xsi:type="sf:Account"
xmlns:sf="urn:sobject.enterprise.soap.sforce.com">
               <sf:Id>00130000002K9TrAAK</sf:Id>
               <sf:AccountNumber>0000000</sf:AccountNumber>
               <sf:Name>New Account Name from Update Sample</sf:Name>
               <sf:Website>www.oz.com</sf:Website>
            </result>
            <result xsi:type="sf:Account"
xmlns:sf="urn:sobject.enterprise.soap.sforce.com">
               <sf:Id>00130000002K9TsAAK</sf:Id>
               <sf:AccountNumber>0000000</sf:AccountNumber>
               <sf:Name>Golden Straw</sf:Name>
               <sf:Website>www.oz.com</sf:Website>
            </result>
         </retrieveResponse>
      </soapenv:Body>
   </soapenv:Envelope>
```

Sample Request Message—retrieve Call—Partner API

```
POST https://na1.salesforce.com/services/Soap/u/5.0 HTTP/1.1
User-Agent: Mozilla/4.0 (compatible; MSIE 6.0; MS Web Services Client Protocol
1.1.4322.573)
Content-Type: text/xml; charset=utf-8
SOAPAction: ""
Content-Length: 752
Expect: 100-continue
Host: na1.salesforce.com

<?xml version="1.0" encoding="utf-8"?>
   <soap:Envelope xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xmlns:xsd="http://
www.w3.org/2001/XMLSchema">
      <soap:Header>
         <QueryOptions xmlns="urn:partner.soap.sforce.com">
            <batchSize>3</batchSize>
         </QueryOptions>
         <SessionHeader xmlns="urn:partner.soap.sforce.com">
            <sessionId>4I1yNabX_5FyOGOnuFO64VpRP0791e_jXlNKOSU</sessionId>
         </SessionHeader>
      </soap:Header>
      <soap:Body>
         <retrieve xmlns="urn:partner.soap.sforce.com">
            <fieldList>Id, AccountNumber, Name, Website</fieldList>
            <sObjectType>Account</sObjectType>
            <ids>00130000002K9QgAAK</ids>
            <ids>00130000002K9QhAAK</ids>
         </retrieve>
      </soap:Body>
   </soap:Envelope>
```


Sample Response Message—retrieve Call—Partner API

```
HTTP/1.1 200 OK
Server: sfdc
Content-Type: text/xml; charset=utf-8
Transfer-Encoding: chunked
Date: Wed, 07 Jul 2004 16:46:22 GMT

<?xml version="1.0" encoding="UTF-8"?>
   <soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
xmlns:xsd="http://www.w3.org/2001/XMLSchema" xmlns:xsi="http://www.w3.org/2001/
XMLSchema-instance">
      <soapenv:Body>
         <retrieveResponse xmlns="urn:partner.soap.sforce.com">
            <result xsi:type="sf:sObject"
xmlns:sf="urn:sobject.partner.soap.sforce.com">
               <sf:type>Account</sf:type>
               <sf:Id>00130000002K9QgAAK</sf:Id>
               <sf:Id>00130000002K9QgAAK</sf:Id>
               <sf:AccountNumber>0000000</sf:AccountNumber>
               <sf:Name>Golden Straw</sf:Name>
               <sf:Website>www.oz.com</sf:Website>
            </result>
            <result xsi:type="sf:sObject"
xmlns:sf="urn:sobject.partner.soap.sforce.com">
               <sf:type>Account</sf:type>
               <sf:Id>00130000002K9QhAAK</sf:Id>
               <sf:Id>00130000002K9QhAAK</sf:Id>
               <sf:AccountNumber>0000000</sf:AccountNumber>
               <sf:Name>Golden Straw</sf:Name>
               <sf:Website>www.oz.com</sf:Website>
            </result>
         </retrieveResponse>
      </soapenv:Body>
   </soapenv:Envelope>
```

Sample SOAP Messages—search

This topic provides the following sample SOAP messages for the search call:

- Sample Request Message—search Call—Enterprise API
- Sample Response Message—search Call—Enterprise API
- Sample Request Message—search Call—Partner API
- Sample Response Message—search Call—Partner API

Sample Request Message—search Call—Enterprise API

```
POST /services/Soap/c/5.0 HTTP/1.0
Content-Type: text/xml; charset=utf-8
Accept: application/soap+xml, application/dime, multipart/related, text/*
User-Agent: Axis/1.1
Host: na1.salesforce.com
Cache-Control: no-cache
Pragma: no-cache
SOAPAction: ""
Content-Length: 818
```



```
<?xml version="1.0" encoding="UTF-8"?>
   <soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
xmlns:xsd="http://www.w3.org/2001/XMLSchema" xmlns:xsi="http://www.w3.org/2001/
XMLSchema-instance">
      <soapenv:Header>
         <ns1:SessionHeader soapenv:mustUnderstand="0"
xmlns:ns1="urn:enterprise.soap.sforce.com">
            <ns2:sessionId
xmlns:ns2="urn:enterprise.soap.sforce.com">uejMEgecf8yidA3jXlNKOSU</
ns2:sessionId>
         </ns1:SessionHeader>
      </soapenv:Header>
      <soapenv:Body>
         <search xmlns="urn:enterprise.soap.sforce.com">
           <searchString>find {4159017000} in phone fields returning contact(id,
phone, firstname, lastname), lead(id, phone, firstname, lastname), account(id,
phone, name)</searchString>
         </search>
      </soapenv:Body>
   </soapenv:Envelope>
```

Sample Response Message—search Call—Enterprise API

```
HTTP/1.0 200 OK
Server: sfdc
Content-Type: text/xml; charset=utf-8
Date: Wed, 07 Jul 2004 18:20:18 GMT

<?xml version="1.0" encoding="UTF-8"?>
   <soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
xmlns:xsd="http://www.w3.org/2001/XMLSchema" xmlns:xsi="http://www.w3.org/2001/
XMLSchema-instance">
      <soapenv:Body>
         <searchResponse xmlns="urn:enterprise.soap.sforce.com">
            <result>
               <searchRecords>
                  <record xsi:type="sf:Contact"
xmlns:sf="urn:sobject.enterprise.soap.sforce.com">
                     <sf:Id>00330000000xEflAAE</sf:Id>
                     <sf:FirstName>Toast</sf:FirstName>
                     <sf:LastName>Barr</sf:LastName>
                     <sf:Phone>(415) 901-7000</sf:Phone>
                  </record>
               </searchRecords>
               <searchRecords>
                  <record xsi:type="sf:Account"
xmlns:sf="urn:sobject.enterprise.soap.sforce.com">
                     <sf:Id>00130000000ZdoSAAS</sf:Id>
                     <sf:Name>Burlington Textiles Corp of America</sf:Name>
                     <sf:Phone>(415) 901-7000</sf:Phone>
                  </record>
               </searchRecords>
               <searchRecords>
                  <record xsi:type="sf:Lead"
xmlns:sf="urn:sobject.enterprise.soap.sforce.com">
                     <sf:Id>00Q30000000asfGEAQ</sf:Id>
                     <sf:FirstName>Kathy</sf:FirstName>
                     <sf:LastName>Snyder</sf:LastName>
                     <sf:Phone>(415) 901-7000</sf:Phone>
```



```
                </record>
             </searchRecords>
          </result>
       </searchResponse>
    </soapenv:Body>
 </soapenv:Envelope>
```

Sample Request Message—search Call—Partner API

```
POST https://na1.salesforce.com/services/Soap/u/5.0 HTTP/1.1
User-Agent: Mozilla/4.0 (compatible; MSIE 6.0; MS Web Services Client Protocol
1.1.4322.573)
Content-Type: text/xml; charset=utf-8
SOAPAction: ""
Content-Length: 688
Expect: 100-continue
Host: na1.salesforce.com

   <?xml version="1.0" encoding="utf-8"?>
      <soap:Envelope xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xmlns:xsd="http://
www.w3.org/2001/XMLSchema">
         <soap:Header>
            <SessionHeader xmlns="urn:partner.soap.sforce.com">
               <sessionId>4I1yNabX_5FyOGOnuFO64VpRP0_jXlNKOSU</sessionId>
            </SessionHeader>
         </soap:Header>
         <soap:Body>
            <search xmlns="urn:partner.soap.sforce.com">
               <searchString>find {4159017000} in phone fields returning
contact(id, phone, firstname, lastname), lead(id, phone, firstname, lastname),
account(id, phone, name)</searchString>
            </search>
         </soap:Body>
      </soap:Envelope>
```

Sample Response Message—search Call—Partner API

```
HTTP/1.1 200 OK
Server: sfdc
Content-Type: text/xml; charset=utf-8
Transfer-Encoding: chunked
Date: Wed, 07 Jul 2004 17:03:03 GMT

   <?xml version="1.0" encoding="UTF-8"?>
     <soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
xmlns:xsd="http://www.w3.org/2001/XMLSchema" xmlns:xsi="http://www.w3.org/2001/
XMLSchema-instance">
         <soapenv:Body>
            <searchResponse xmlns="urn:partner.soap.sforce.com">
               <result>
                  <searchRecords>
                     <record xsi:type="sf:sObject"
xmlns:sf="urn:sobject.partner.soap.sforce.com">
                        <sf:type>Contact</sf:type>
                        <sf:Id>00330000000xEflAAE</sf:Id>
                        <sf:Id>00330000000xEflAAE</sf:Id>
```



```
                        <sf:Phone>(415) 901-7000</sf:Phone>
                        <sf:FirstName>Toast</sf:FirstName>
                        <sf:LastName>Barr</sf:LastName>
                     </record>
                  </searchRecords>
                  <searchRecords>
                     <record xsi:type="sf:sObject"
xmlns:sf="urn:sobject.partner.soap.sforce.com">
                        <sf:type>Account</sf:type>
                        <sf:Id>00130000000ZdoSAAS</sf:Id>
                        <sf:Id>00130000000ZdoSAAS</sf:Id>
                        <sf:Phone>(415) 901-7000</sf:Phone>
                       <sf:Name>Burlington Textiles Corp of America</sf:Name>
                     </record>
                  </searchRecords>
                  <searchRecords>
                     <record xsi:type="sf:sObject"
xmlns:sf="urn:sobject.partner.soap.sforce.com">
                        <sf:type>Lead</sf:type>
                        <sf:Id>00Q30000000asfGEAQ</sf:Id>
                        <sf:Id>00Q30000000asfGEAQ</sf:Id>
                        <sf:Phone>(415) 901-7000</sf:Phone>
                        <sf:FirstName>Kathy</sf:FirstName>
                        <sf:LastName>Snyder</sf:LastName>
                     </record>
                  </searchRecords>
               </result>
            </searchResponse>
         </soapenv:Body>
      </soapenv:Envelope>
```

Sample SOAP Messages—setPassword

This topic provides the following sample SOAP messages for the setPassword call:

- Sample Request Message—setPassword Call—Enterprise API
- Sample Response Message—setPassword Call—Enterprise API
- Sample Request Message—setPassword Call—Partner API
- Sample Response Message—setPassword Call—Partner API

Sample Request Message—setPassword Call—Enterprise API

```
POST /services/Soap/c/5.0 HTTP/1.0
Content-Type: text/xml; charset=utf-8
Accept: application/soap+xml, application/dime, multipart/related, text/*
User-Agent: Axis/1.1
Host: na1.salesforce.com
Cache-Control: no-cache
Pragma: no-cache
SOAPAction: ""
Content-Length: 716

<?xml version="1.0" encoding="UTF-8"?>
   <soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
xmlns:xsd="http://www.w3.org/2001/XMLSchema" xmlns:xsi="http://www.w3.org/2001/
XMLSchema-instance">
      <soapenv:Header>
```



```
         <ns1:SessionHeader soapenv:mustUnderstand="0"
xmlns:ns1="urn:enterprise.soap.sforce.com">
            <ns2:sessionId
xmlns:ns2="urn:enterprise.soap.sforce.com">uejMEgecf8yidA3zCFeT6bakjXlNKOSU</
ns2:sessionId>
         </ns1:SessionHeader>
      </soapenv:Header>
      <soapenv:Body>
         <setPassword xmlns="urn:enterprise.soap.sforce.com">
            <userId>005300000003e43AAI</userId>
            <password>newsecret</password>
         </setPassword>
      </soapenv:Body>
   </soapenv:Envelope>
```

Sample Response Message—setPassword Call—Enterprise API

```
HTTP/1.0 200 OK
Server: sfdc
Content-Type: text/xml; charset=utf-8
Date: Wed, 07 Jul 2004 18:18:52 GMT

<?xml version="1.0" encoding="UTF-8"?>
   <soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
xmlns:xsd="http://www.w3.org/2001/XMLSchema" xmlns:xsi="http://www.w3.org/2001/
XMLSchema-instance">
      <soapenv:Body>
         <setPasswordResponse xmlns="urn:enterprise.soap.sforce.com">
            <result/>
         </setPasswordResponse>
      </soapenv:Body>
   </soapenv:Envelope>
```

Sample Request Message—setPassword Call—Partner API

```
POST https://na1.salesforce.com/services/Soap/u/5.0 HTTP/1.1
User-Agent: Mozilla/4.0 (compatible; MSIE 6.0; MS Web Services Client Protocol
1.1.4322.573)
Content-Type: text/xml; charset=utf-8
SOAPAction: ""
Content-Length: 582
Expect: 100-continue
Host: na1.salesforce.com

   <?xml version="1.0" encoding="utf-8"?>
      <soap:Envelope xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xmlns:xsd="http://
www.w3.org/2001/XMLSchema">
         <soap:Header>
            <SessionHeader xmlns="urn:partner.soap.sforce.com">
               <sessionId>4I1yNabX_5FyOGOnuFO64VpRP0_jXlNKOSU</sessionId>
            </SessionHeader>
         </soap:Header>
         <soap:Body>
            <setPassword xmlns="urn:partner.soap.sforce.com">
               <userId>00530000000ii63AAI</userId>
               <password>dave</password>
```



```
            </setPassword>
         </soap:Body>
      </soap:Envelope>
```

Sample Response Message—setPassword Call—Partner API

```
HTTP/1.1 200 OK
Server: sfdc
Content-Type: text/xml; charset=utf-8
Transfer-Encoding: chunked
Date: Wed, 07 Jul 2004 16:55:55 GMT

   <?xml version="1.0" encoding="UTF-8"?>
     <soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
xmlns:xsd="http://www.w3.org/2001/XMLSchema" xmlns:xsi="http://www.w3.org/2001/
XMLSchema-instance">
         <soapenv:Body>
            <setPasswordResponse xmlns="urn:partner.soap.sforce.com">
               <result/>
            </setPasswordResponse>
         </soapenv:Body>
      </soapenv:Envelope>
```

Sample SOAP Messages—update

This topic provides the following sample SOAP messages for the update call:

- Sample Request Message—update Call—Enterprise API
- Sample Response Message—update Call—Enterprise API
- Sample Request Message—update Call—Partner API
- Sample Response Message—update Call—Partner API

Sample Request Message—update Call—Enterprise API

```
POST /services/Soap/c/5.0 HTTP/1.0
Content-Type: text/xml; charset=utf-8
Accept: application/soap+xml, application/dime, multipart/related, text/*
User-Agent: Axis/1.1
Host: na1.salesforce.com
Cache-Control: no-cache
Pragma: no-cache
SOAPAction: ""
Content-Length: 1022

<?xml version="1.0" encoding="UTF-8"?>
   <soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
xmlns:xsd="http://www.w3.org/2001/XMLSchema" xmlns:xsi="http://www.w3.org/2001/
XMLSchema-instance">
      <soapenv:Header>
         <ns1:SessionHeader soapenv:mustUnderstand="0"
xmlns:ns1="urn:enterprise.soap.sforce.com">
            <ns2:sessionId
xmlns:ns2="urn:enterprise.soap.sforce.com">uejMEgecf8yidA3zCFeT6baks1vDk8jUjXlNKO
SU</ns2:sessionId>
         </ns1:SessionHeader>
      </soapenv:Header>
```



```
      <soapenv:Body>
         <update xmlns="urn:enterprise.soap.sforce.com">
            <sObjects xsi:type="ns3:Account"
xmlns:ns3="urn:sobject.enterprise.soap.sforce.com">
               <ns3:Id>00130000002K9TrAAK</ns3:Id>
               <ns3:Name>New Account Name from Update Sample</ns3:Name>
            </sObjects>
            <sObjects xsi:type="ns4:Account"
xmlns:ns4="urn:sobject.enterprise.soap.sforce.com">
               <ns4:fieldsToNull>Name</ns4:fieldsToNull>
               <ns4:Id>SLFKJLFKJ</ns4:Id>
            </sObjects>
         </update>
      </soapenv:Body>
   </soapenv:Envelope>
```

Sample Response Message—update Call—Enterprise API

```
HTTP/1.0 200 OK
Server: sfdc
Content-Type: text/xml; charset=utf-8
Date: Wed, 07 Jul 2004 18:14:38 GMT

<?xml version="1.0" encoding="UTF-8"?>
   <soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
xmlns:xsd="http://www.w3.org/2001/XMLSchema" xmlns:xsi="http://www.w3.org/2001/
XMLSchema-instance">
      <soapenv:Body>
         <updateResponse xmlns="urn:enterprise.soap.sforce.com">
            <result>
               <errors xsi:nil="true"/>
               <id>00130000002K9TrAAK</id>
               <success>true</success>
            </result>
            <result>
               <errors>
                  <fields xsi:nil="true"/>
                  <message>bad id SLFKJLFKJ</message>
                  <statusCode>MALFORMED_ID</statusCode>
               </errors>
               <id xsi:nil="true"/>
               <success>false</success>
            </result>
         </updateResponse>
      </soapenv:Body>
   </soapenv:Envelope>
```

Sample Request Message—update Call—Partner API

```
POST https://na1.salesforce.com/services/Soap/u/5.0 HTTP/1.1
User-Agent: Mozilla/4.0 (compatible; MSIE 6.0; MS Web Services Client Protocol
1.1.4322.573)
Content-Type: text/xml; charset=utf-8
SOAPAction: ""
Content-Length: 978
Expect: 100-continue
Host: na1.salesforce.com
```



```
   <?xml version="1.0" encoding="utf-8"?>
      <soap:Envelope xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xmlns:xsd="http://
www.w3.org/2001/XMLSchema">
         <soap:Header>
            <SessionHeader xmlns="urn:partner.soap.sforce.com">
               <sessionId>4I1yNabX_5FyOGOnuFO64VpRP0791eQNk_jXlNKOSU</sessionId>
            </SessionHeader>
         </soap:Header>
         <soap:Body>
            <update xmlns="urn:partner.soap.sforce.com">
               <sObjects>
                  <type xmlns="urn:sobject.partner.soap.sforce.com">Account</type>
                   <Id
xmlns="urn:sobject.partner.soap.sforce.com">00130000002K9QgAAK</Id>
                   <Name xmlns="">New Account Name from Update Sample</Name>
               </sObjects>
               <sObjects>
                  <type xmlns="urn:sobject.partner.soap.sforce.com">Account</type>
                  <fieldsToNull xmlns="urn:sobject.partner.soap.sforce.com">Name</
fieldsToNull>
                  <Id xmlns="urn:sobject.partner.soap.sforce.com">S:DLFKJLFKJ</Id>
                   <Name xmlns="">Error</Name>
               </sObjects>
            </update>
         </soap:Body>
      </soap:Envelope>
```

Sample Response Message—update Call—Partner API

```
HTTP/1.1 200 OK
Server: sfdc
Content-Type: text/xml; charset=utf-8
Transfer-Encoding: chunked
Date: Wed, 07 Jul 2004 16:48:07 GMT

   <?xml version="1.0" encoding="UTF-8"?>
     <soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
xmlns:xsd="http://www.w3.org/2001/XMLSchema" xmlns:xsi="http://www.w3.org/2001/
XMLSchema-instance">
         <soapenv:Body>
            <updateResponse xmlns="urn:partner.soap.sforce.com">
               <result>
                  <errors xsi:nil="true"/>
                  <id>00130000002K9QgAAK</id>
                  <success>true</success>
               </result>
               <result>
                  <errors>
                     <fields xsi:nil="true"/>
                     <message>bad id S:DLFKJLFKJ</message>
                     <statusCode>MALFORMED_ID</statusCode>
                  </errors>
                  <id xsi:nil="true"/>
                  <success>false</success>
               </result>
            </updateResponse>
         </soapenv:Body>
```



```
    </soapenv:Envelope>
```


CHAPTER 10: Primitive Data Types

The sforce API uses the following primitive data types:

Table 63: Primitive Data Types Used in the sforce API

Value	Description
`xsd:base64Binary`	Base 64-encoded binary data.
`xsd:boolean`	Boolean (True / false) values.
`xsd:date`	Date values.
`xsd:dateTime`	Date/time values (timestamps).
`xsd:double`	Double values.
`xsd:int`	Integer values.
`xsd:string`	Character strings.

These data types are used in the SOAP messages that are exchanged between your client application and the sforce Web service. When writing your client application, you simply 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.

The primitive data types are:

- specified in the World Wide Web Consortium's publication *XML Schema Part 2: Datatypes* at the following URL: http://www.w3.org/TR/xmlschema-2/
- enumerated in the SOAPType field of the Field type, which is described in the `fields` property of the DescribeSObjectResult.

Primitive types are used as a standardized way to define, send, receive, and interpret basic data types in the SOAP messages exchanged between client applications and the sforce Web service. In addition, primitive data types are interpreted in a salesforce.com-specific way, which is useful for display formatting and for numeric conversion (adding values of different currencies).

For example, salesforce.com chooses to interpret a double value passed via SOAP (as an `xsd:double`) in a number of possible ways, depending on the field definition. If the field type for that data is currency, salesforce.com handles the display of the data by prepending it with a currency symbol and inserting a decimal for precision. Similarly, if the field type is percent, salesforce.com handles the display of the data by appending a percent sign (%). Regardless of the field type, however, the value is sent in the SOAP message as a double.

CHAPTER 11: Other Concepts

This topic describes advanced but optional topics that might be of interest to some readers. It contains the following sections:

INTERNATIONALIZATION AND CHARACTER SETS

The salesforce.com server supports either full Unicode characters or ISO-8859-1 characters. The character set for your organization depends on the salesforce.com instance your organization uses. If your organization logs into the ssl.salesforce.com, then your encoding is ISO-8859-1. All other instances use UTF-8. You can determine the character set for your organization by calling describeGlobal and inspecting the `encoding` value returned in the DescribeGlobalResult.

If your organization uses ISO-8859-1 encoding, then all data sent to the sforce Web service must be encoded in ISO-8859-1. Characters outside the valid ISO-8859-1 range might be truncated or cause an error.

Note

The Web service response is encoded in the character set used by your organization (UTF-8 or ISO-8859-1). Either way, the encoded data is usually handled for you by the SOAP client.

XML COMPLIANCE

The sforce API is based on XML, which requires all documents to be well formed. Part of that requirement is that certain Unicode characters are not allowed in an XML document, even in an escaped form, and that others must be encoded according to their location. Normally this is handled for you by any standard SOAP or XML client. Clients must be able to parse any normal XML escape sequence, and must not pass up invalid XML characters.

Some characters, as mentioned, are illegal even if they are escaped. The illegal characters include the Unicode surrogate blocks and a few other Unicode characters. All are seldom-used control characters that are usually not important in any data, and tend to cause problems with many programs. Although they are not allowed in XML documents, they are allowed in HTML documents and may be present in sforce data. The illegal characters will be stripped from any API response.

The following characters are illegal:

Table 64: Illegal XML Characters

0xFFFE
0xFFFF

Table 64: Illegal XML Characters

Control characters 0x0 - 0x19 (Not including 0x9, 0xA, 0xD, tab, newline, and carriage return)
0xD800 - 0xDFFF

For UTF-8 encoding, sforce supports only the basic UCS-2 plane and does not support any of the extended UCS-4 characters. UCS-4 support is extremely rare in any system. UCS-2 is the set that Java and Windows NT support. For more information about XML characters and character sets, see: `www.w3.org/TR/REC-xml#charsets`.

COMPRESSION

The sforce API allows the use of compression on the request and the response, using the standards defined by the HTTP 1.1 specification. This is automatically supported by some SOAP/WSDL clients, and can be manually added to others. Check the sforce.com site for more information on particular clients.

Compression is not used unless the client specifically indicates that it supports compression. For better performance, we suggest that clients accept and support compression as defined by the HTTP 1.1 specification.

To indicate that the client supports compression, you should include the HTTP header "Accept-Encoding: gzip, deflate" or a similar heading. The server compresses the response if the client properly specifies this header. The response includes the header "Content-Encoding: deflate" or "Content-Encoding: gzip," as appropriate. You can also compress any request by including a "Content-Encoding: deflate" or "gzip" header.

Most clients are partially constrained by their network connection, even on a corporate LAN. The sforce API allows the use of compression to improve performance. Almost all clients can benefit from response compression, and many clients may benefit from compression of requests as well. The sforce server supports deflate and gzip compression according the HTTP 1.1 specification.

Response Compression

The sforce server can optionally compress responses. Responses are compressed only if the client sends an Accept-Encoding header with either gzip or deflate compression specified. The server is not required to compress the response even if you have specified Accept-Encoding, but it normally does. If the server compresses the response, it also specifies a Content-Encoding header with the name of the compression algorithm used, either gzip or deflate.

Request Compression

Clients can also compress requests. The sforce server decompresses any requests before processing. The client must send up a Content-Encoding HTTP header with the name of the appropriate compression algorithm. For more information, see:

- Content-Encoding at: `www.w3.org/Protocols/rfc2616/rfc2616-sec14.html#sec14.11`
- Accept-Encoding at: `www.w3.org/Protocols/rfc2616/rfc2616-sec14.html#sec14.3`
- Content Codings at: `www.w3.org/Protocols/rfc2616/rfc2616-sec3.html#sec3.5`

MULTIPLE INSTANCE SUPPORT

The sforce API provides access to all worldwide sforce servers, including:

- US/North America 0
- US/North America 1
- EMEA/Europe
- AP/Japan

The different sets of servers are referred to as *instances*.

Note

An organization is not guaranteed to be on a particular instance, just because that organization is located in a particular region, although that is generally true. Also, additional instances may be added in the future.

The SOAP implementation of the sforce API also provides a single login server. You can log in to any organization via a single entry point, without having to hard-code the instance for your organization. To access an organization via the sforce API, you must first authenticate the session by sending a login request to the login server at the following URL:

```
https://www.salesforce.com/services/Soap/c/5.0
```

The insecure version of the URL is also supported:

```
http://www.salesforce.com/services/Soap/c/5.0
```

All subsequent calls to the server during the session should be made to the URL returned in the login response.

HTTP Persistent Connections

Most clients achieve better performance if they use HTTP 1.1 persistent connection to reuse the socket connection for multiple requests. Persistent connections are normally handled by your SOAP/WSDL client automatically. For more details, see the HTTP 1.1 specification at:

```
www.w3.org/Protocols/rfc2616/rfc2616-sec8.html#sec8.1
```

HTTP Chunking

Clients that use HTTP 1.1 may receive chunked responses. Chunking is normally handled by your SOAP/WSDL client automatically.

Index

Q

R

S

T

U

W