CONTENTS

Chapter 1: Introducing the Place Order REST API ............................................. 1
  Requirements and Limitations .......................................................................... 1
  Understanding Authentication ............................................................................ 2
    Defining Connected Apps ............................................................................... 2
    Understanding OAuth Endpoints ..................................................................... 3
  Understanding the Web Server OAuth Authentication Flow ......................... 3
  Understanding the User-Agent OAuth Authentication Flow .......................... 9
  Understanding the Username-Password OAuth Authentication Flow ............. 12
  Understanding the OAuth Refresh Token Process ......................................... 14
  Finding Additional Resources ......................................................................... 16

Chapter 2: Understanding Place Order REST API Resources ............................ 17
  Add a Contract and Orders to an Existing Account ....................................... 17
  Add an Order to an Existing Account ............................................................... 19
  Add Orders to an Existing Contract ................................................................. 20
  Add Order Products to an Existing Order ...................................................... 23
  Get Details About a Contract ........................................................................... 24
  Get Details About an Order ............................................................................ 26
  Filter Details About a Contract ....................................................................... 27
  Filter Details About an Order ........................................................................... 28

Chapter 3: Resource Reference ........................................................................... 30
  Create Contract-based Orders ......................................................................... 30
  Contract-based Orders ..................................................................................... 33
  Create Order .................................................................................................... 36
  Order ................................................................................................................. 38

Index ................................................................................................................... 42
The Salesforce Place Order REST API is a composite API that gives programmatic access to contract, order, and order product data, as well as child custom object data of contracts and orders in Salesforce. With this composite API, you can create contract, order, order product, and custom object records in a single call. The Place Order REST API can be used by any organization that has orders and API enabled.

Use the Place Order REST API, a REST-based composite application programming interface, to:

- Add orders to a new or existing contract, and add order products to those orders.
- Add order products to a new or existing order.
- Add custom objects to a new or existing contract or order.
- Retrieve order records under a given contract, as well as those orders’ custom objects and order products.
- Retrieve order product records under a given order, as well as custom object records under the order and its order products.
- Retrieve a filtered list of orders under a given contract or order products under a given order.

Requirements and Limitations

To access the Place Order REST API, you must establish a secure OAuth session ID.

Consider these limitations and general limits when using the Place Order REST API.

Limits and Limitations

- 2000 records per request or the API maximum limit for your organization—whichever is lower.
- Responses and requests are in JSON.
  When using Place Order REST API resources that require a request or response body, use Content-Type: application/json.
- Each call can only affect one top-level entity.
  For orders under a contract, you need one call for each new or existing contract you’re adding orders, order products, or custom objects to. For orders not under a contract, you need one call for each new or existing order you’re adding order products or custom objects to.
- In each resource, you can create custom objects at a depth of one level below the top-level entity.
  - /services/data/latest API version/commerce/sale supports custom object records on contracts and orders.
  - /services/data/latest API version/commerce/sale/order supports custom object records on orders and order products.
- Custom objects are not supported as children of other custom objects.
- To filter GET results, query parameters must be a fully qualified field name. The parent entity must be lower-cased (such as order), and the field must match the defined relationship name (such as orders.StatusCode).
  For example, to get a list of all orders with a draft status under a given contract, you must use contract.orders.StatusCode='Draft'.
- When you create a new order, the Status Code must be Draft and the Status must be any value that corresponds to a Status Code of Draft.
You can’t update existing records.

Understanding Authentication

Salesforce uses the OAuth protocol to allow users of applications to securely access data without having to reveal username and password credentials.

Before making REST API calls, you must authenticate the application user using OAuth 2.0. To do so, you’ll need to:

- Set up your application as a connected app in the Salesforce organization.
- Determine the correct Salesforce OAuth endpoint for your connected app to use.
- Authenticate the connected app user via one of several different OAuth 2.0 authentication flows. An OAuth authentication flow defines a series of steps used to coordinate the authentication process between your application and Salesforce. Supported OAuth flows include:
  - Web server flow, where the server can securely protect the consumer secret.
  - User-agent flow, used by applications that cannot securely store the consumer secret.
  - Username-password flow, where the application has direct access to user credentials.

After successfully authenticating the connected app user with Salesforce, you’ll receive an access token which can be used to make authenticated REST API calls.

Defining Connected Apps

To authenticate using OAuth, you must create a connected app that defines your application’s OAuth settings for the Salesforce organization.

When you develop an external application that needs to authenticate with Salesforce, you need to define it as a new connected app within the Salesforce organization that informs Salesforce of this new authentication entry point.

Use the following steps to create a new connected app.

1. From Setup, enter Apps in the Quick Find box, then select Apps and click New to start defining a connected app.
2. Enter the name of your application.
3. Enter the contact email information, as well as any other information appropriate for your application.
4. Select Enable OAuth Settings.
5. Enter a Callback URL. Depending on which OAuth flow you use, this is typically the URL that a user’s browser is redirected to after successful authentication. As this URL is used for some OAuth flows to pass an access token, the URL must use secure HTTP (HTTPS) or a custom URI scheme.
6. Add all supported OAuth scopes to Selected OAuth Scopes. These scopes refer to permissions given by the user running the connected app.
7. Enter a URL for Info URL. This is where the user can go for more information about your application.
8. Click Save. The Consumer Key is created and displayed, and the Consumer Secret is created (click the link to reveal it). Once you define a connected app, you use the consumer key and consumer secret to authenticate your application. See Creating a Connected App in the Salesforce online help for specific steps to create a connected app for the type of authentication you need.
Understanding OAuth Endpoints

OAuth endpoints are the URLs you use to make OAuth authentication requests to Salesforce.

You need to use the correct Salesforce OAuth endpoint when issuing authentication requests in your application. The primary OAuth endpoints are:

- For authorization: https://login.salesforce.com/services/oauth2/authorize
- For token requests: https://login.salesforce.com/services/oauth2/token
- For revoking OAuth tokens: https://login.salesforce.com/services/oauth2/revoke

All endpoints require secure HTTP (HTTPS). Each OAuth flow defines which endpoints you need to use and what request data you need to provide.

If you’re verifying authentication on a sandbox organization, use “test.salesforce.com” instead of “login.salesforce.com” in all the OAuth endpoints listed above.

Understanding the Web Server OAuth Authentication Flow

The Web server authentication flow is used by applications that are hosted on a secure server. A critical aspect of the Web server flow is that the server must be able to protect the consumer secret. You can also use code challenge and verifier values in the flow to prevent authorization code interception.

In this flow, the client application requests the authorization server to redirect the user to another web server or resource that authorizes the user and sends the application an authorization code. The application uses the authorization code to request an access token. The following shows the steps for this flow.
1. The application redirects the user to the appropriate Salesforce authorization endpoint, such as https://login.salesforce.com/services/oauth2/authorize. The following parameters are required:

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>response_type</td>
<td>Must be code for this authentication flow.</td>
</tr>
<tr>
<td>client_id</td>
<td>The Consumer Key from the connected app definition.</td>
</tr>
<tr>
<td>redirect_uri</td>
<td>The Callback URL from the connected app definition.</td>
</tr>
</tbody>
</table>

The following parameters are optional:

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>code_challenge</td>
<td>Specifies the SHA256 hash value of the code_verifier value in the token request to help prevent authorization code interception attacks. The hash value must be base64url encoded as defined here: <a href="https://tools.ietf.org/html/rfc4648#section-5">https://tools.ietf.org/html/rfc4648#section-5</a>.</td>
</tr>
<tr>
<td>Parameter</td>
<td>Description</td>
</tr>
<tr>
<td>-----------</td>
<td>-------------</td>
</tr>
</tbody>
</table>
| display   | Changes the login page’s display type. Valid values are:  
  - page—Full-page authorization screen. This is the default value if none is specified.  
  - popup—Compact dialog optimized for modern Web browser popup windows.  
  - touch—Mobile-optimized dialog designed for modern smartphones such as Android and iPhone.  
  - mobile—Mobile optimized dialog designed for smartphones such as BlackBerry OS 5 that don’t support touch screens. |
| immediate | Determines whether the user should be prompted for login and approval. Values are either true or false. Default is false.  
  - If set to true, and if the user is currently logged in and has previously approved the application, the approval step is skipped.  
  - If set to true and the user is not logged in or has not previously approved the application, the session is immediately terminated with the immediate_unsuccessful error code. |
<p>| login_hint| Provides a valid username value to pre-populate the login page with the username. For example: login_hint=<a href="mailto:username@company.com">username@company.com</a>. If a user already has an active session in the browser, then the login_hint parameter does nothing; the active user session continues. |
| nonce     | Specifies a value to be returned in the response; this is useful for detecting “replay” attacks. Optional with the openid scope for getting a user ID token. |</p>
<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
</table>
| prompt    | Specifies how the authorization server prompts the user for reauthentication and reapproval. This parameter is optional. The only values Salesforce supports are:  
  • login—The authorization server must prompt the user for reauthentication, forcing the user to log in again.  
  • consent—The authorization server must prompt the user for reapproval before returning information to the client.  
It is valid to pass both values, separated by a space, to require the user to both log in and reauthorize. For example:  
?prompt=login%20consent |
| scope     | Specifies what data your application can access. See “Scope Parameter Values” in the online help for more information. |
| state     | Specifies any additional URL-encoded state data to be returned in the callback URL after approval. |

An example authorization URL might look something like the following:


2. The user logs into Salesforce with their credentials. The user is interacting with the authorization endpoint directly, so the application never sees the user’s credentials. After successfully logging in, the user is asked to authorize the application. Note that if the user has already authorized the application, this step is skipped.

3. After Salesforce confirms that the client application is authorized, the end-user’s Web browser is redirected to the callback URL specified by the redirect_uri parameter. Salesforce appends authorization information to the redirect URL with the following values:

<table>
<thead>
<tr>
<th>Parameters</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>code</td>
<td>Authorization code the consumer must use to obtain the access and refresh tokens.</td>
</tr>
<tr>
<td>state</td>
<td>The state value that was passed in as part of the initial request, if applicable.</td>
</tr>
</tbody>
</table>

An example callback URL with authorization information might look something like:

https://www.mysite.com/authcode_callback?code=aWekysIEeqM9PiT6Ef0Cnr6MoLIfwWyRJccQqRdP8f9IINokharAS09IA7UNP6RiVScerFhc4w%3D%3D
4. The application extracts the authorization code and passes it in a request to Salesforce for an access token. This request is a POST request sent to the appropriate Salesforce token request endpoint, such as https://login.salesforce.com/services/oauth2/token. The following parameters are required:

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>grant_type</td>
<td>Value must be authorization_code for this flow.</td>
</tr>
<tr>
<td>client_id</td>
<td>The Consumer Key from the connected app definition.</td>
</tr>
<tr>
<td>client_secret</td>
<td>The Consumer Secret from the connected app definition.</td>
</tr>
<tr>
<td>redirect_uri</td>
<td>The Callback URL from the connected app definition.</td>
</tr>
<tr>
<td>code</td>
<td>Authorization code the consumer must use to obtain the access and refresh tokens.</td>
</tr>
</tbody>
</table>

The following parameters are optional:

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>client_assertion</td>
<td>Instead of passing in client_secret you can choose to provide a client_assertion and client_assertion_type. If a client_secret parameter is not provided, Salesforce checks for the client_assertion and client_assertion_type automatically. The value of client_assertion must be a typical JWT bearer token, signed with the private key associated with the OAuth consumer’s uploaded certificate. Only the RS256 algorithm is currently supported. For more information on using client_assertion, see the OpenID Connect specifications for the private_key_jwt client authentication method.</td>
</tr>
<tr>
<td>client_assertion_type</td>
<td>Provide this value when using the client_assertion parameter. The value of client_assertion_type must be urn:ietf:params:oauth:client-assertion-type:jwt-bearer.</td>
</tr>
<tr>
<td>code_verifier</td>
<td>Specifies 128 bytes of random data with high enough entropy to make it difficult to guess the value to help prevent authorization code interception attacks. The value also must be base64url encoded as defined here: <a href="https://tools.ietf.org/html/rfc4648#section-5">https://tools.ietf.org/html/rfc4648#section-5</a>.</td>
</tr>
</tbody>
</table>

- If the code_verifier value is provided in the token request and a code_challenge value is in the authorization request, Salesforce compares the code_verifier to the code_challenge. If the code_verifier is invalid or doesn’t match, the login fails with the invalid_grant error code.
- If the code_verifier value is provided in the token request, but a code_challenge value was not provided
in the authorization request, the login fails with the `invalid_grant` error code.

<table>
<thead>
<tr>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>The value should be base64url-encoded only once.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>format</td>
<td>Expected return format. The default is <code>json</code>. Values are:</td>
</tr>
<tr>
<td></td>
<td>• <code>urlencoded</code></td>
</tr>
<tr>
<td></td>
<td>• <code>json</code></td>
</tr>
<tr>
<td></td>
<td>• <code>xml</code></td>
</tr>
<tr>
<td></td>
<td>The return format can also be specified in the header of the request using one of the following:</td>
</tr>
<tr>
<td></td>
<td>• <code>Accept: application/x-www-form-urlencoded</code></td>
</tr>
<tr>
<td></td>
<td>• <code>Accept: application/json</code></td>
</tr>
<tr>
<td></td>
<td>• <code>Accept: application/xml</code></td>
</tr>
</tbody>
</table>

An example access token POST request might look something like:

```
POST /services/oauth2/token HTTP/1.1
Host: login.salesforce.com
grant_type=authorization_code&code=aPrxsmIEeqM9PiQroGEWx1UiMqd95_5JUZVEhsOFhS8EVvbYBBJli2W5fn3zbo.8hojaNW_1g%3D%3D&client_id=3MVG9lKcPoNI
VBIPJjdwlJ9LLM82HnFVVX19KY1uA5mu0QqEWhqKpoW3svG3XHrXDjCQjK1mdgAvhCs
CA9GE&client_secret=1955279925675241571&
```

5. If this request is successful, the server returns a response body that contains the following:

<table>
<thead>
<tr>
<th>Parameters</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>access_token</code></td>
<td>Access token that acts as a session ID that the application uses for making requests. This token should be protected as though it were user credentials.</td>
</tr>
<tr>
<td><code>refresh_token</code></td>
<td>Token that can be used in the future to obtain new access tokens.</td>
</tr>
<tr>
<td></td>
<td><strong>Warning:</strong> This value is a secret. You should treat it like the user's password and use appropriate measures to protect it.</td>
</tr>
<tr>
<td><code>instance_url</code></td>
<td>Identifies the Salesforce instance to which API calls should be sent.</td>
</tr>
<tr>
<td><code>id</code></td>
<td>Identity URL that can be used to both identify the user as well as query for more information about the user. Can be used in an HTTP request to get more information about the end user.</td>
</tr>
</tbody>
</table>
Introducing the Place Order REST API

<table>
<thead>
<tr>
<th>Parameters</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>issued_at</td>
<td>When the signature was created, represented as the number of seconds since the Unix epoch (00:00:00 UTC on 1 January 1970).</td>
</tr>
<tr>
<td>signature</td>
<td>Base64-encoded HMAC-SHA256 signature signed with the consumer's private key containing the concatenated ID and issued_at value. The signature can be used to verify that the identity URL wasn't modified because it was sent by the server.</td>
</tr>
</tbody>
</table>

An example JSON response body might look something like:

```json
{"id":"https://login.salesforce.com/id/00Dx0000000BV7z/005x00000012Q9P",
 "issued_at":"1278448101416",
 "refresh_token":"5Aep8614iLM.Dq661ePDmPEgaAW9Oh_L3JKkDpB4xReb54_p2ebnUG0h65b4KUVDPntWEofWM39yg==",
 "instance_url":"https://na1.salesforce.com",
 "signature":"CMJ4l+CCaPQiKjoOEwEig9H4wqhppULShk4J2urAe+fVg=",
 "access_token":"00Dx0000000BV7z!AR8AQP0jITN80ESej5EbaZTFG0RNBaT1cyWk7TrqoDjoNIWQ2ME_sT2zBjfmOE6zMHq6y8FIW4eWze9JksNEkWU1.Cju7m4"}
```

6. The application uses the provided access token and refresh token to access protected user data.

Understanding the User-Agent OAuth Authentication Flow

The user-agent authentication flow is used by client applications (consumers) residing in the user’s device. This could be implemented in a browser using a scripting language such as JavaScript, or from a mobile device or a desktop application. These consumers cannot keep the client secret confidential.

In this flow, the client application requests the authorization server to redirect the user to another Web server or resource which is capable of extracting the access token and passing it back to the application. The following shows the steps for this flow.

---

9
1. The application redirects the user to the appropriate Salesforce authorization endpoint, such as https://login.salesforce.com/services/oauth2/authorize. The following parameters are required:

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>response_type</td>
<td>Must be <code>token</code> for this authentication flow</td>
</tr>
<tr>
<td>client_id</td>
<td>The <code>Consumer Key</code> from the connected app definition.</td>
</tr>
<tr>
<td>redirect_uri</td>
<td>The <code>Callback URL</code> from the connected app definition.</td>
</tr>
</tbody>
</table>

The following parameters are optional:

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>display</td>
<td>Changes the login page's display type. Valid values are:</td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>• page—Full-page authorization screen. This is the default value if none is specified.</td>
</tr>
<tr>
<td></td>
<td>• popup—Compact dialog optimized for modern Web browser popup windows.</td>
</tr>
<tr>
<td></td>
<td>• touch—Mobile-optimized dialog designed for modern smartphones such as Android and iPhone.</td>
</tr>
</tbody>
</table>
### Parameter Description

- **mobile**—Mobile optimized dialog designed for smartphones such as BlackBerry OS 5 that don’t support touch screens.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>scope</td>
<td>Specifies what data your application can access. See “Scope Parameter Values” in the online help for more information.</td>
</tr>
<tr>
<td>state</td>
<td>Specifies any additional URL-encoded state data to be returned in the callback URL after approval.</td>
</tr>
</tbody>
</table>

An example authorization URL might look something like the following:

```plaintext
https://login.salesforce.com/services/oauth2/authorize?response_type=token&
client_id=3MVG91KcPoNINVBIPJjdW1J9LLbP_pgwoJYyuisjQhr_LLurNDv7AgQvDTZwCoZuDZxPcmBv4o.8ds.5iE&redirect_uri=https%3A%2F%2Fwww.mysite.com%2Fuser_callback.jsp&
state=mystate
```

2. The user logs into Salesforce with their credentials. The user interacts with the authorization endpoint directly, so the application never sees the user’s credentials.

3. Once authorization is granted, the authorization endpoint redirects the user to the redirect URL. This URL is defined in the remote access application created for the application. Salesforce appends access token information to the redirect URL with the following values:

<table>
<thead>
<tr>
<th>Parameters</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>access_token</td>
<td>Access token that acts as a session ID that the application uses for making requests. This token should be protected as though it were user credentials.</td>
</tr>
<tr>
<td>expires_in</td>
<td>Amount of time the access token is valid, in seconds.</td>
</tr>
<tr>
<td>refresh_token</td>
<td>Token that can be used in the future to obtain new access tokens.</td>
</tr>
<tr>
<td></td>
<td><strong>Warning:</strong> This value is a secret. You should treat it like the user’s password and use appropriate measures to protect it.</td>
</tr>
<tr>
<td>state</td>
<td>The state value that was passed in as part of the initial request, if applicable.</td>
</tr>
<tr>
<td>instance_url</td>
<td>Identifies the Salesforce instance to which API calls should be sent.</td>
</tr>
<tr>
<td>id</td>
<td>Identity URL that can be used to both identify the user as well as query for more information about the user. Can be used in an HTTP request to get more information about the end user.</td>
</tr>
<tr>
<td>Parameters</td>
<td>Description</td>
</tr>
<tr>
<td>------------</td>
<td>-------------</td>
</tr>
<tr>
<td>issued_at</td>
<td>When the signature was created, represented as the number of seconds since the Unix epoch (00:00:00 UTC on 1 January 1970).</td>
</tr>
<tr>
<td>signature</td>
<td>Base64-encoded HMAC-SHA256 signature signed with the consumer’s private key containing the concatenated ID and issued_at value. The signature can be used to verify that the identity URL wasn’t modified because it was sent by the server.</td>
</tr>
</tbody>
</table>

An example callback URL with access information appended after the hash sign (#) might look something like:

```plaintext
https://www.mysite.com/user_callback.jsp#access_token=00Dx0000000BV7z%21AR8AQBMB8J_xr9kLqmZIRyQxZgLcM4HV41aGtW0qW3JCzf5xdTGGSSoVim8FFJkZEqxbJaFbberKGk8v8AnYrvChG4qJbQo8&refresh_token=5Aep8614iLM.Dq661ePdmPEgaAW9Oh_L3JKkDpB4xReb54_pZVtildPEk8aimw4Hr9ne7VXXVSIQ%3D%3D&expires_in=7200&state=mystate
```

4. The application uses the provided access token and refresh token to access protected user data.

Keep the following considerations in mind when using the user-agent OAuth flow:

- Because the access token is encoded into the redirection URI, it might be exposed to the end-user and other applications residing on the computer or device. If you’re authenticating using JavaScript, call `window.location.replace();` to remove the callback from the browser’s history.

**Understanding the Username-Password OAuth Authentication Flow**

The username-password authentication flow can be used to authenticate when the consumer already has the user’s credentials.

In this flow, the user’s credentials are used by the application to request an access token as shown in the following steps.

⚠️ **Warning:** This OAuth authentication flow involves passing the user’s credentials back and forth. Use this authentication flow only when necessary. No refresh token will be issued.
1. The application uses the user’s username and password to request an access token. This is done via an out-of-band POST request to the appropriate Salesforce token request endpoint, such as https://login.salesforce.com/services/oauth2/token. The following request fields are required:

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>grant_type</td>
<td>Must be password for this authentication flow.</td>
</tr>
<tr>
<td>client_id</td>
<td>The Consumer Key from the connected app definition.</td>
</tr>
<tr>
<td>client_secret</td>
<td>The Consumer Secret from the connected app definition.</td>
</tr>
<tr>
<td>username</td>
<td>End-user’s username.</td>
</tr>
<tr>
<td>password</td>
<td>End-user’s password.</td>
</tr>
</tbody>
</table>

**Note:** You must append the user’s security token to their password. A security token is an automatically-generated key from Salesforce. For example, if a user’s password is mypassword, and their security token is XXXXXXXXXX, then the value provided for this parameter must be mypasswordXXXXXXXXX. For more information on security tokens see “Reset Your Security Token” in the online help.
An example request body might look something like the following:

```
grant_type=password&client_id=3MVG9lKcPoNINVBIPJjdw1J9LLM82HnFVVX19KY1uA5mu0QqEWhqKpoW3svG3XHrXDiCQjK1mdgAvhCscA9GE&client_secret=1955279925675241571&username=testuser%40salesforce.com&password=mypassword123456
```

2. Salesforce verifies the user credentials, and if successful, sends a response to the application with the access token. This response contains the following values:

<table>
<thead>
<tr>
<th>Parameters</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>access_token</td>
<td>Access token that acts as a session ID that the application uses for making requests. This token should be protected as though it were user credentials.</td>
</tr>
<tr>
<td>instance_url</td>
<td>Identifies the Salesforce instance to which API calls should be sent.</td>
</tr>
<tr>
<td>id</td>
<td>Identity URL that can be used to both identify the user as well as query for more information about the user. Can be used in an HTTP request to get more information about the end user.</td>
</tr>
<tr>
<td>issued_at</td>
<td>When the signature was created, represented as the number of seconds since the Unix epoch (00:00:00 UTC on 1 January 1970).</td>
</tr>
<tr>
<td>signature</td>
<td>Base64-encoded HMAC-SHA256 signature signed with the consumer’s private key containing the concatenated ID and issued_at value. The signature can be used to verify that the identity URL wasn’t modified because it was sent by the server.</td>
</tr>
</tbody>
</table>

An example response body might look something like:

```
{"id":"https://login.salesforce.com/id/00Dx0000000BV7z/005x00000012Q9P","issued_at":"1278448832702","instance_url":"https://na1.salesforce.com","signature":"0CmxinZir53Yex7nE0TD+zMpvIWYGb/bdJh6XfOH6EQ=","access_token":"00Dx0000000BV7z!AR8AQAxo9UfVkh8A1VOGomt9Czx9LjHnSSpwBmmbRcgKFMwOtVxjTrkWl9ye6PE3Ds1eQz3z8jr3W7_VbWmEu4Q8TVGSTHxs"}
```

3. The application uses the provided access token to access protected user data.

Keep the following considerations in mind when using the user-agent OAuth flow:

- Since the user is never redirected to login at Salesforce in this flow, the user can’t directly authorize the application, so no refresh tokens can be used. If your application requires refresh tokens, you should consider using the Web server or user-agent OAuth flow.

**Understanding the OAuth Refresh Token Process**

The Web server OAuth authentication flow and user-agent flow both provide a refresh token that can be used to obtain a new access token.

Access tokens have a limited lifetime specified by the session timeout in Salesforce. If an application uses an expired access token, a “Session expired or invalid” error is returned. If the application is using the Web server or user-agent OAuth authentication flows, a refresh token may be provided during authorization that can be used to get a new access token.
The client application obtains a new access token by sending a POST request to the token request endpoint with the following request parameters:

<table>
<thead>
<tr>
<th>Parameters</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>grant_type</td>
<td>Value must be refresh_token.</td>
</tr>
<tr>
<td>refresh_token</td>
<td>The refresh token the client application already received.</td>
</tr>
<tr>
<td>client_id</td>
<td>The Consumer Key from the connected app definition.</td>
</tr>
<tr>
<td>client_secret</td>
<td>The Consumer Secret from the connected app definition. This parameter is optional.</td>
</tr>
<tr>
<td>format</td>
<td>Expected return format. The default is json. Values are:</td>
</tr>
<tr>
<td></td>
<td>• urllencoded</td>
</tr>
<tr>
<td></td>
<td>• json</td>
</tr>
<tr>
<td></td>
<td>• xml</td>
</tr>
<tr>
<td></td>
<td>The return format can also be specified in the header of the request using one of the following:</td>
</tr>
<tr>
<td></td>
<td>• Accept: application/x-www-form-urlencoded</td>
</tr>
<tr>
<td></td>
<td>• Accept: application/json</td>
</tr>
<tr>
<td></td>
<td>• Accept: application/xml</td>
</tr>
<tr>
<td></td>
<td>This parameter is optional.</td>
</tr>
</tbody>
</table>

An example refresh token POST request might look something like:

```plaintext
POST /services/oauth2/token HTTP/1.1
Host: https://login.salesforce.com/
grant_type=refresh_token&client_id=3MVG91KcPoNINVB1PjidwLJ9LLM82HnFVVX19KY1uA5mu0QqEWhqKpoW3svG3XhrXDiCQjk1mgAvhCscA9GE&client_secret=1955279925675241571&refresh_token=your token here
```

Once Salesforce verifies the refresh token request, it sends a response to the application with the following response body parameters:

<table>
<thead>
<tr>
<th>Parameters</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>access_token</td>
<td>Access token that acts as a session ID that the application uses for making requests. This token should be protected as though it were user credentials.</td>
</tr>
<tr>
<td>instance_url</td>
<td>Identifies the Salesforce instance to which API calls should be sent.</td>
</tr>
<tr>
<td>id</td>
<td>Identity URL that can be used to both identify the user as well as query for more information about the user. Can be used in an HTTP request to get more information about the end user.</td>
</tr>
<tr>
<td>issued_at</td>
<td>When the signature was created, represented as the number of seconds since the Unix epoch (00:00:00 UTC on 1 January 1970).</td>
</tr>
</tbody>
</table>
An example JSON response body might look something like:

```
{
    "id": "https://login.salesforce.com/id/00Dx0000000BV7z/005x00000012Q9P",
    "issued_at": "1278448384422",
    "instance_url": "https://na1.salesforce.com",
    "signature": "SSSbLO/gBhmmyNUvN180DBFYHzakxOMggYtu+hDPsc=",
    "access_token": "00Dx0000000BV7z!AR8AQP0jITN80ESeaj5Eba2TFG0RNBA1cyWk7T
rqoDjoNIWQ2ME_sTZzBjfmOE6zMHq6y8PlW4eWze9JksNEkWU1.Cju7m4"
}
```

Keep in mind the following considerations when using the refresh token OAuth process:

- The session timeout for an access token can be configured in Salesforce from Setup by entering Session Settings in the Quick Find box, then selecting Session Settings.
- If the application uses the username-password OAuth authentication flow, no refresh token is issued, as the user cannot authorize the application in this flow. If the access token expires, the application using username-password OAuth flow must re-authenticate the user.

**Finding Additional Resources**

The following resources provide additional information about using OAuth with Salesforce:

- **Authenticating Apps with OAuth**
- **Digging Deeper into OAuth on Force.com**
- **Using OAuth to Authorize External Applications**

The following resources are examples of third party client libraries that implement OAuth that you might find useful:

- For Ruby on Rails: OmniAuth
- For Java: Apache Amber
- Additional OAuth client libraries: OAuth.net
The Place Order REST API is designed to let you integrate your order creation system easily.

Use this API to:

**IN THIS SECTION:**
- Add a Contract and Orders to an Existing Account
- Add an Order to an Existing Account
- Add Orders to an Existing Contract
- Add Order Products to an Existing Order
- Get Details About a Contract
- Get Details About an Order
- Filter Details About a Contract
- Filter Details About an Order

**Add a Contract and Orders to an Existing Account**

Here’s an example of a POST request using the Create Contract-based Orders resource to create a contract—with child orders, order products, and custom objects—to an existing account.

**Example usage**

```
/services/data/v30.0/commerce/sale
```

**Example request body**

```json
{
  "contract": [
    {
      "attributes": {
        "type": "Contract"
      },
      "AccountId": "001D000000JRDAv",
      "StartDate": "2013-08-01",
      "Status": "Draft",
      "ContractTerm": "1",
      "Test_Contract1__r": {
        "attributes": {
          "type": "Test_Contract1__c"
        },
        "Name": "Contract Custom Object"
      }
    }
  ]
}
```
Example JSON response body

```json
{
  "totalSize": 1,
  "done": true,
  "records": [
    {
      "attributes": {
        "type": "Contract",
        "url": "services/data/v30.0/sobjects/Contract/800D0000000PIcMIAW"
      },
      "Id": "800D0000000PIcMIAW",
      "Orders": {
        "totalSize": 1,
        "done": true,
        "records": [
          {
            "attributes": {
              "type": "Order",
              "url": "services/data/data/v30.0/sobjects/Order/801D0000000G0ylIAC"
            },
            "Id": "801D0000000G0ylIAC",
            "OrderItems": {
              "totalSize": 1,
              "done": true,
              "records": [
                {
                  "attributes": {
```
Add an Order to an Existing Account

Here's an example of a POST request using the Create Order resource to create an order with order products for an existing account.

**Example usage**

```
/services/data/v30.0/commerce/sale/order
```

**Example request body**

```json
{
  "order": [
    {
      "attributes": {
        "type": "Order"
      },
      "EffectiveDate": "2013-07-11",
      "Status": "Draft",
      "billingCity": "SFO-Inside-OrderEntity-1",
      "accountId": "001D000000JRDAv",
      "Pricebook2Id": "01sD0000000G2NjIAK",
      "OrderItems": {
        "records": [
          {
            "attributes": {
              "type": "OrderItem"
            },
            "PricebookEntryId": "01uD0000001c5toIAA",
            "quantity": "1",
            "UnitPrice": "15.99"
          }
        ]
      }
    }
  ]
}
```
Example JSON response body

```
{
  "totalSize": 1,
  "done": true,
  "records": [
    {
      "attributes": {
        "type": "Order",
        "url": "/services/data/v30.0/sobjects/Order/801D0000000G0ySIAS"
      },
      "Id": "801D0000000G0ySIAS",
      "OrderItems": {
        "totalSize": 1,
        "done": true,
        "records": [
          {
            "attributes": {
              "type": "OrderItem",
              "url": "/services/data/v30.0/sobjects/OrderItem/802D0000000CKp8IAG"
            },
            "Id": "802D0000000CKp8IAG"
          }
        ]
      }
    }
  ]
}
```

Add Orders to an Existing Contract

Here's an example of a PATCH request using the Contract-based Orders resource to add new orders and order products to an existing contract.

**Example usage**

```
/services/data/v30.0/commerce/sale/800D0000000PFL8IAO
```

**Example request body**

```
{
  "contract": {
    "attributes": {
      "type": "Contract"
    },
    "Id": "800D0000000PFL8IAO",
    "Orders": {
      "records": [
        {
          "attributes": {
            "type": "Order"
          }
        }
      ]
    }
  }
}
```
Add Orders to an Existing Contract

Understanding Place Order REST API Resources
Example JSON response body

```json
{
  "totalSize": 1,
  "done": true,
  "records": [
    {
      "attributes": {
        "type": "Contract"
      },
      "Orders": {
        "totalSize": 2,
        "done": true,
        "records": [
          {
            "attributes": {
              "type": "Order",
              "url": "/services/data/v29.0/sobjects/Order/801D0000000G0xsIAC"
            },
            "Id": "801D0000000G0xsIAC",
            "OrderItems": {
              "totalSize": 2,
              "done": true,
              "records": [
                {
                  "attributes": {
                    "type": "OrderItem",
                    "url": "/services/data/v29.0/sobjects/OrderItem/802D0000000CKoPIAW"
                  },
                  "Id": "802D0000000CKoPIAW"
                },
                {
                  "attributes": {
                    "type": "OrderItem",
                    "url": "/services/data/v29.0/sobjects/OrderItem/802D0000000CKoQIAW"
                  },
                  "Id": "802D0000000CKoQIAW"
                }
              ]
            }
          }
        ]
      }
    }
  ]
}
```
Add Order Products to an Existing Order

Here's an example of a PATCH request using the Order resource to add order products to an existing order.

**Example usage**

```
/services/data/v30.0/commerce/sale/order/801D0000000Frh8
```

**Example request body**

```
{
  "order": [
    {
      "attributes": {
        "type": "Order"
      },
      "Id": "801D0000000Frh8",
      "OrderItems": {
        "records": [
          {
            "attributes": {
              "type": "OrderItem"
            },
            "PricebookEntryId": "01uD0000001cCd1",
            "Id": "802D0000000CKoRIAW"
          },
          {
            "attributes": {
              "type": "OrderItem"
            },
            "PricebookEntryId": "01uD0000001cCd2",
            "Id": "802D0000000CKoRIAW"
          },
          {
            "attributes": {
              "type": "OrderItem"
            },
            "PricebookEntryId": "01uD0000001cCd3",
            "Id": "802D0000000CKoRIAW"
          }
        ]
      }
    }
  ]
}
```
Example JSON response body

```json
{
  "totalSize": 1,
  "done": true,
  "records": [
    {
      "attributes": {
        "type": "Order"
      },
      "OrderItems": {
        "totalSize": 2,
        "done": true,
        "records": [
          {
            "attributes": {
              "type": "OrderItem",
              "url": "/services/data/v30.0/sobjects/OrderItem/802D0000000CJtMIAW"
            },
            "Id": "802D0000000CJtMIAW"
          },
          {
            "attributes": {
              "type": "OrderItem",
              "url": "/services/data/v30.0/sobjects/OrderItem/802D0000000CJtNIAW"
            },
            "Id": "802D0000000CJtNIAW"
          }
        ]
      }
    }
  ]
}
```

Get Details About a Contract

Here's an example of a GET request using the Contract-based Orders resource to query details about a contract and its child orders, order products, and custom objects.
Example usage
/services/data/v30.0/commerce/sale/800D0000000PFHp

Example request body
None

Example JSON response body
{
  "totalSize" : 1,
  "done" : true,
  "records" : [ {
    "attributes" : {
      "type" : "Contract",
      "url" : "/services/data/v30.0/sobjects/Contract/800D0000000PFHpIAO"
    },
    "Id" : "800D0000000PFHpIAO",
    "Orders" : [ {
      "totalSize" : 4,
      "done" : true,
      "records" : [ {
        "attributes" : {
          "type" : "Order",
          "url" : "/services/data/v30.0/sobjects/Order/801D0000000FqzsIAC"
        },
        "Id" : "801D0000000FqzsIAC",
        "OrderItems" : [ {
          "totalSize" : 3,
          "done" : true,
          "records" : [ {
            "attributes" : {
              "type" : "OrderItem",
              "url" : "/services/data/v30.0/sobjects/OrderItem/802D0000000CJX0IAO"
            },
            "Id" : "802D0000000CJX0IAO"
          }, {
            "attributes" : {
              "type" : "OrderItem",
              "url" : "/services/data/v30.0/sobjects/OrderItem/802D0000000CJYDIA4"
            },
            "Id" : "802D0000000CJYDIA4"
          }, {
            "attributes" : {
              "type" : "OrderItem",
              "url" : "/services/data/v30.0/sobjects/OrderItem/802D0000000CKFCIA4"
            },
            "Id" : "802D0000000CKFCIA4"
          } ]
        } ]
      },
      "Custom_Objects__r" : null
    },
    "attributes" : {
      "type" : "Order",
      "url" : "/services/data/v30.0/sobjects/Order/801D0000000FpNEIA0"
    }
  } ]
}
Get Details About an Order

Here's an example of a GET request using the Order resource to query details about an order and its order products and custom object records.
Filter Details About a Contract

Here's an example of a GET request using the Contract-based Orders resource to query a given contract's activated orders.

Example usage
/services/data/v30.0/commerce/sale/800D0000000PFL8?contract.orders.StatusCode='Activated'

Example request body
None

Example JSON response body
{
  "totalSize": 1,
  "done": true,
  "records": [
    {
      "attributes": {
        "type": "Order",
        "url": "/services/data/v30.0/sobjects/Order/801D0000000FzsMlAS"
      },
      "Id": "801D0000000FzsMlAS",
      "OrderItems": {
        "totalSize": 2,
        "done": true,
        "records": [
          {
            "attributes": {
              "type": "OrderItem",
              "url": "/services/data/v30.0/sobjects/OrderItem/802D0000000CKIHIA4"
            },
            "Id": "802D0000000CKIHIA4"
          },
          {
            "attributes": {
              "type": "OrderItem",
              "url": "/services/data/v30.0/sobjects/OrderItem/802D0000000CKIGIA4"
            },
            "Id": "802D0000000CKIGIA4"
          }
        ]
      },
      "Custom_Objects__r": null
    }
  ]
}
Filter Details About an Order

Here’s an example of a GET request using the Order resource to query details for order products with a certain start date for a given order.

Example usage

/services/data/v30.0/commerce/sale/order/801D0000000FzsM?order.orderItems.effectiveDate=2013-08-05
Example request body
None

Example JSON response body

```
{
  "totalSize": 1,
  "done": true,
  "records": [{
    "attributes": {
      "type": "Order",
      "url": "/services/data/v30.0/sobjects/Order/801D0000000FzsMIAS"
    },
    "Id": "801D0000000FzsMIAS",
    "OrderItems": {
      "totalSize": 1,
      "done": true,
      "records": [{
        "attributes": {
          "type": "OrderItem",
          "url": "/services/data/v30.0/sobjects/OrderItem/802D0000000CKIHIA4"
        },
        "Id": "802D0000000CKIHIA4"
      }]
    },
    "Custom_Objects__r": null
  }]
}
```
Each Place Order REST API resource is a URI used with an HTTP method (such as GET). Use these resources to integrate reporting directly into your applications.

Resources for the Place Order REST API are:

<table>
<thead>
<tr>
<th>Resource</th>
<th>Supported HTTP Method</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>/services/data/latest API version/commerce/sale</td>
<td>POST</td>
<td>Add new orders, order products, and custom objects to a new contract.</td>
</tr>
<tr>
<td>/services/data/latest API version/commerce/sale/contract ID</td>
<td>PATCH, GET</td>
<td>Add new orders, order products, and custom objects to an existing contract. Retrieve a contract’s child orders, order products, and custom objects.</td>
</tr>
<tr>
<td>/services/data/latest API version/commerce/sale/order</td>
<td>POST</td>
<td>Add new order products and custom objects to a new order.</td>
</tr>
<tr>
<td>/services/data/latest API version/commerce/sale/order/order ID</td>
<td>PATCH, GET</td>
<td>Add new order products and custom objects to an existing order. Retrieve an order’s child order products and custom objects.</td>
</tr>
</tbody>
</table>

**Create Contract-based Orders**

With this resource, you can create a new contract with orders and order products, as well as custom object records on the contract or order level.

**Syntax**

**URI**

/services/data/latest API version/commerce/sale

Available since release

30.0

**Formats**

JSON

**HTTP methods**

POST
## Request body

<table>
<thead>
<tr>
<th>Property</th>
<th>Type</th>
<th>Description</th>
<th>Since Version</th>
</tr>
</thead>
<tbody>
<tr>
<td>attributes</td>
<td>Attributes</td>
<td>Type and URL of the new contract.</td>
<td>30.0</td>
</tr>
<tr>
<td>AccountId</td>
<td>String</td>
<td>Unique record identifier for the parent account.</td>
<td>30.0</td>
</tr>
<tr>
<td>Status</td>
<td>String</td>
<td>Status of the contract.</td>
<td>30.0</td>
</tr>
<tr>
<td>Orders</td>
<td>Orders[]</td>
<td>Child orders of the new contract.</td>
<td>30.0</td>
</tr>
<tr>
<td>CustomObject__r</td>
<td>Custom Objects[]</td>
<td>Child custom object records of the new contract.</td>
<td>30.0</td>
</tr>
</tbody>
</table>

### Attributes

<table>
<thead>
<tr>
<th>Property</th>
<th>Type</th>
<th>Description</th>
<th>Since Version</th>
</tr>
</thead>
<tbody>
<tr>
<td>type</td>
<td>String</td>
<td>Format of the resource.</td>
<td>30.0</td>
</tr>
</tbody>
</table>

### Orders

<table>
<thead>
<tr>
<th>Property</th>
<th>Type</th>
<th>Description</th>
<th>Since Version</th>
</tr>
</thead>
<tbody>
<tr>
<td>attributes</td>
<td>Attributes</td>
<td>Type and URL of the new order record.</td>
<td>30.0</td>
</tr>
<tr>
<td>Status</td>
<td>String</td>
<td>Status of the order.</td>
<td>30.0</td>
</tr>
<tr>
<td>Pricebook2Id</td>
<td>String</td>
<td>Unique record identifier for the associated price book.</td>
<td>30.0</td>
</tr>
<tr>
<td>OrderItems</td>
<td>Order Products[]</td>
<td>Child order products of the new order.</td>
<td>30.0</td>
</tr>
<tr>
<td>CustomObject__r</td>
<td>Custom Objects[]</td>
<td>Child custom object records of the new order.</td>
<td>30.0</td>
</tr>
</tbody>
</table>

### Custom Object Records

<table>
<thead>
<tr>
<th>Property</th>
<th>Type</th>
<th>Description</th>
<th>Since Version</th>
</tr>
</thead>
<tbody>
<tr>
<td>attributes</td>
<td>Attributes</td>
<td>Type and URL of the custom object record.</td>
<td>30.0</td>
</tr>
<tr>
<td>Id</td>
<td>String</td>
<td>Unique record identifier.</td>
<td>30.0</td>
</tr>
</tbody>
</table>
Order Products

<table>
<thead>
<tr>
<th>Property</th>
<th>Type</th>
<th>Description</th>
<th>Since Version</th>
</tr>
</thead>
<tbody>
<tr>
<td>attributes</td>
<td>Attributes</td>
<td>Type and URL of the order product record.</td>
<td>30.0</td>
</tr>
<tr>
<td>PricebookEntryId</td>
<td>String</td>
<td>Unique record identifier for the associated price book entry.</td>
<td>30.0</td>
</tr>
<tr>
<td>quantity</td>
<td>Number</td>
<td>Number of units of the order product.</td>
<td>30.0</td>
</tr>
<tr>
<td>UnitPrice</td>
<td>Currency</td>
<td>The unit price for the order product.</td>
<td>30.0</td>
</tr>
</tbody>
</table>

Request parameters
None

Response body

<table>
<thead>
<tr>
<th>Property</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>totalSize</td>
<td>Number</td>
<td>Total number of records added.</td>
</tr>
<tr>
<td>done</td>
<td>Boolean</td>
<td>When true, the operation was successful.</td>
</tr>
<tr>
<td>records</td>
<td>Records[]</td>
<td>Attributes and IDs of records.</td>
</tr>
</tbody>
</table>

Records

<table>
<thead>
<tr>
<th>Property</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>attributes</td>
<td>Attributes</td>
<td>Type and URL for the new record.</td>
</tr>
<tr>
<td>Id</td>
<td>String</td>
<td>Unique identifier for the new record.</td>
</tr>
</tbody>
</table>

Attributes

<table>
<thead>
<tr>
<th>Property</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>type</td>
<td>String</td>
<td>Format of the resource.</td>
</tr>
<tr>
<td>url</td>
<td>String</td>
<td>Resource URL.</td>
</tr>
</tbody>
</table>

Example

See Add a Contract and Orders to an Existing Account on page 17.

Usage

You can only create one contract at a time. You can’t POST new orders onto an existing contract. For that, see the Contract-based Orders resource.
Contract-based Orders

With this resource, you can add one or more new orders to an existing contract, as well as order products and custom object records for each order, or you can retrieve data for a specific contract.

If available, GET method retrieves the contract’s child orders and order products, as well as custom objects under the contract and orders.

Syntax

URI

/services/data/latest API version/commerce/sale/contractId

For retrieving filtered data:

/services/data/latest API version/commerce/sale/contract ID?contract.orders.field name=value

Available since release

30.0

Formats

JSON

HTTP methods

PATCH, GET

Request parameters

- You can use parameters for all standard and custom fields on contracts, orders, order products, and any custom objects directly related to these objects.
- The parameters must be fully qualified. For example: objectname.relationshipname.fieldname=value
  - Object name must be all lower-case.
  - Relationship name must match what is defined on the object and is case-sensitive.
  - Field name isn’t case sensitive.
  - If the value is a string, it must be encased in single quotation marks. If the value is a number, it must not be encased. If the value is a date, it should be in the YYYY-MM-DD format.

- You can use multiple parameter fields, separated by "&", to make more detailed filters. For example:

/services/data/v30.0/commerce/sale/{contractId}?contract.status='Activated' &contract.Orders.status='Draft'&contract.Orders.OrderItems.unitprice=300

The following aren’t supported:

- Arrays of values. For example: contract.orders.Status='Activated','Draft'.
- Operators: >, >=, < and <=
- The OR condition
### Parameters

<table>
<thead>
<tr>
<th>Property</th>
<th>Description</th>
<th>Since Version</th>
</tr>
</thead>
<tbody>
<tr>
<td>contract</td>
<td>The object name of the record being filtered. In this resource, this is always contract.</td>
<td>30.0</td>
</tr>
<tr>
<td>orders</td>
<td>The relationship name of the field that the order’s data will be filtered by. In this resource, this is always orders.</td>
<td>30.0</td>
</tr>
<tr>
<td>field name</td>
<td>The field whose value to filter by. For example, if you want to only retrieve orders with a status category, the field name is Status.</td>
<td>30.0</td>
</tr>
<tr>
<td>value</td>
<td>The value to filter by. For example, if you want to only retrieve orders with a status category of Activated, the value is Activated.</td>
<td>30.0</td>
</tr>
</tbody>
</table>

### Request body

<table>
<thead>
<tr>
<th>Property</th>
<th>Type</th>
<th>Description</th>
<th>Since Version</th>
</tr>
</thead>
<tbody>
<tr>
<td>attributes</td>
<td>Attributes</td>
<td>Type of the contract.</td>
<td>30.0</td>
</tr>
<tr>
<td>Id</td>
<td>String</td>
<td>Unique contract identifier.</td>
<td>30.0</td>
</tr>
<tr>
<td>Orders</td>
<td>Orders[]</td>
<td>Child orders of the contract.</td>
<td>30.0</td>
</tr>
</tbody>
</table>

#### Attributes

<table>
<thead>
<tr>
<th>Property</th>
<th>Type</th>
<th>Description</th>
<th>Since Version</th>
</tr>
</thead>
<tbody>
<tr>
<td>type</td>
<td>String</td>
<td>Format of the resource.</td>
<td>30.0</td>
</tr>
</tbody>
</table>

#### Order Records

<table>
<thead>
<tr>
<th>Property</th>
<th>Type</th>
<th>Description</th>
<th>Since Version</th>
</tr>
</thead>
<tbody>
<tr>
<td>attributes</td>
<td>Attributes</td>
<td>Type of the order record.</td>
<td>30.0</td>
</tr>
<tr>
<td>Status</td>
<td>String</td>
<td>Status of the order.</td>
<td>30.0</td>
</tr>
<tr>
<td>contractId</td>
<td>String</td>
<td>Unique record identifier for the parent contract.</td>
<td>30.0</td>
</tr>
<tr>
<td>pricebook2Id</td>
<td>String</td>
<td>Unique record identifier for the associated price book.</td>
<td>30.0</td>
</tr>
<tr>
<td>OrderItems</td>
<td>Order Products[]</td>
<td>Child order products of the order.</td>
<td>30.0</td>
</tr>
<tr>
<td>CustomObject__r</td>
<td>Custom Objects[]</td>
<td>Child custom object records of the order.</td>
<td>30.0</td>
</tr>
</tbody>
</table>
### Order Products

<table>
<thead>
<tr>
<th>Property</th>
<th>Type</th>
<th>Description</th>
<th>Since Version</th>
</tr>
</thead>
<tbody>
<tr>
<td>attributes</td>
<td>Attributes</td>
<td>Type and URL of the order product.</td>
<td>30.0</td>
</tr>
<tr>
<td>PricebookEntryId</td>
<td>String</td>
<td>Unique record identifier for the associated price book entry.</td>
<td>30.0</td>
</tr>
<tr>
<td>quantity</td>
<td>Number</td>
<td>Number of units of the order product.</td>
<td>30.0</td>
</tr>
<tr>
<td>UnitPrice</td>
<td>Currency</td>
<td>The unit price for the order product.</td>
<td>30.0</td>
</tr>
</tbody>
</table>

### Custom Objects

<table>
<thead>
<tr>
<th>Property</th>
<th>Type</th>
<th>Description</th>
<th>Since Version</th>
</tr>
</thead>
<tbody>
<tr>
<td>attributes</td>
<td>Attributes</td>
<td>Type and URL of the custom object.</td>
<td>30.0</td>
</tr>
<tr>
<td>Id</td>
<td>String</td>
<td>Unique custom object identifier.</td>
<td>30.0</td>
</tr>
</tbody>
</table>

### Response body

<table>
<thead>
<tr>
<th>Property</th>
<th>Type</th>
<th>Description</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>totalSize</td>
<td>Number</td>
<td>Total number of records retrieved.</td>
<td></td>
</tr>
<tr>
<td>done</td>
<td>Boolean</td>
<td>When true, the operation was successful.</td>
<td></td>
</tr>
<tr>
<td>records</td>
<td>Records[]</td>
<td>Attributes and ID of contract record.</td>
<td></td>
</tr>
</tbody>
</table>

### Records

<table>
<thead>
<tr>
<th>Property</th>
<th>Type</th>
<th>Description</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>attributes</td>
<td>Attributes</td>
<td>Type and URL of the record.</td>
<td></td>
</tr>
<tr>
<td>Id</td>
<td>String</td>
<td>Unique contract identifier.</td>
<td></td>
</tr>
</tbody>
</table>

### Attributes

<table>
<thead>
<tr>
<th>Property</th>
<th>Type</th>
<th>Description</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>type</td>
<td>String</td>
<td>Format of the resource.</td>
<td></td>
</tr>
<tr>
<td>url</td>
<td>String</td>
<td>Resource URL.</td>
<td></td>
</tr>
</tbody>
</table>
Examples

• Add Orders to an Existing Contract on page 20
• Get Details About a Contract on page 24
• Filter Details About a Contract on page 27

Create Order

With this resource, you can create a new order with order products and custom objects.
If you don’t want to add the order to a contract, you can add it directly to an account. You can only create one new order per call. The request body must have either an account or a contract as its parent record, and it must have a reference to a price book.

Syntax

URI

/services/data/latest API version/commerce/sale/order

Available since release
30.0

Formats
JSON

HTTP methods
POST

Request body

<table>
<thead>
<tr>
<th>Property</th>
<th>Type</th>
<th>Description</th>
<th>Since Version</th>
</tr>
</thead>
<tbody>
<tr>
<td>attributes</td>
<td>Attributes</td>
<td>Type and URL of the new order.</td>
<td>30.0</td>
</tr>
<tr>
<td>OrderItems</td>
<td>Order Products[]</td>
<td>Child order products of the new order.</td>
<td>30.0</td>
</tr>
<tr>
<td>CustomObject__r</td>
<td>Custom Objects[]</td>
<td>Child custom object records of the new order.</td>
<td>30.0</td>
</tr>
</tbody>
</table>

Attributes

<table>
<thead>
<tr>
<th>Property</th>
<th>Type</th>
<th>Description</th>
<th>Since Version</th>
</tr>
</thead>
<tbody>
<tr>
<td>type</td>
<td>String</td>
<td>Format of the resource.</td>
<td>30.0</td>
</tr>
<tr>
<td>url</td>
<td>String</td>
<td>Resource URL.</td>
<td>30.0</td>
</tr>
</tbody>
</table>
### Order Products

<table>
<thead>
<tr>
<th>Property</th>
<th>Type</th>
<th>Description</th>
<th>Since Version</th>
</tr>
</thead>
<tbody>
<tr>
<td>attributes</td>
<td>Attributes</td>
<td>Type and URL of the order product.</td>
<td>30.0</td>
</tr>
<tr>
<td>PricebookEntryId</td>
<td>String</td>
<td>Unique record identifier for the associated price book entry.</td>
<td>30.0</td>
</tr>
<tr>
<td>quantity</td>
<td>Number</td>
<td>Number of units of the order product.</td>
<td>30.0</td>
</tr>
<tr>
<td>UnitPrice</td>
<td>Currency</td>
<td>The unit price for the order product.</td>
<td>30.0</td>
</tr>
<tr>
<td>Id</td>
<td>String</td>
<td>Unique order product identifier.</td>
<td>30.0</td>
</tr>
</tbody>
</table>

### Custom Objects

<table>
<thead>
<tr>
<th>Property</th>
<th>Type</th>
<th>Description</th>
<th>Since Version</th>
</tr>
</thead>
<tbody>
<tr>
<td>attributes</td>
<td>Attributes</td>
<td>Type and URL of the custom object.</td>
<td>30.0</td>
</tr>
<tr>
<td>Id</td>
<td>String</td>
<td>Unique custom object identifier.</td>
<td>30.0</td>
</tr>
</tbody>
</table>

**Request parameters**

None

**Response body**

<table>
<thead>
<tr>
<th>Property</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>totalSize</td>
<td>Number</td>
<td>Total number of records retrieved.</td>
</tr>
<tr>
<td>done</td>
<td>Boolean</td>
<td>When true, the operation was successful.</td>
</tr>
<tr>
<td>records</td>
<td>Records[]</td>
<td>Attributes and ID of contract record.</td>
</tr>
</tbody>
</table>

### Records

<table>
<thead>
<tr>
<th>Property</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>attributes</td>
<td>Attributes on page 37</td>
<td>Type and URL of the record.</td>
</tr>
<tr>
<td>Id</td>
<td>String</td>
<td>Unique contract identifier.</td>
</tr>
</tbody>
</table>

### Attributes

<table>
<thead>
<tr>
<th>Property</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>type</td>
<td>String</td>
<td>Format of the resource.</td>
</tr>
</tbody>
</table>
Resource Reference

**Order**

Use this resource to add one or more new order products and custom object records to an existing order or to retrieve data for a specific order.

You can only PATCH one order at a time.

If available, GET method retrieves the orders' child order products and custom objects under the order or order products.

### Syntax

**URI**

```
/services/data/latest API version/commerce/sale/order/order ID
```

For retrieving filtered data:

```
/services/data/latest API version/commerce/sale/order/order ID?order.orderItems.field name=value
```

**Available since release**

30.0

**Formats**

JSON

**HTTP methods**

POST

**Request body**

<table>
<thead>
<tr>
<th>Property</th>
<th>Type</th>
<th>Description</th>
<th>Since Version</th>
</tr>
</thead>
<tbody>
<tr>
<td>attributes</td>
<td>Attributes</td>
<td>Type and URL of the order.</td>
<td>30.0</td>
</tr>
<tr>
<td>OrderItems</td>
<td>Order Products</td>
<td>Child order products of the new order.</td>
<td>30.0</td>
</tr>
<tr>
<td><code>CustomObject__r</code></td>
<td>Custom Object</td>
<td>Child custom object records of the new order.</td>
<td>30.0</td>
</tr>
</tbody>
</table>

**Example**

See Add an Order to an Existing Account on page 19.
Attributes

<table>
<thead>
<tr>
<th>Property</th>
<th>Type</th>
<th>Description</th>
<th>Since Version</th>
</tr>
</thead>
<tbody>
<tr>
<td>type</td>
<td>String</td>
<td>Format of the resource.</td>
<td>30.0</td>
</tr>
</tbody>
</table>

Order Products

<table>
<thead>
<tr>
<th>Property</th>
<th>Type</th>
<th>Description</th>
<th>Since Version</th>
</tr>
</thead>
<tbody>
<tr>
<td>attributes</td>
<td>Attributes</td>
<td>Type and URL of the order product.</td>
<td>30.0</td>
</tr>
<tr>
<td>PricebookEntryId</td>
<td>String</td>
<td>Unique record identifier for the associated price book entry.</td>
<td>30.0</td>
</tr>
<tr>
<td>quantity</td>
<td>Number</td>
<td>Number of units of the order product.</td>
<td>30.0</td>
</tr>
<tr>
<td>UnitPrice</td>
<td>Currency</td>
<td>The unit price for the order product.</td>
<td>30.0</td>
</tr>
<tr>
<td>orderId</td>
<td>String</td>
<td>Unique record identifier for the parent order.</td>
<td>30.0</td>
</tr>
<tr>
<td>Id</td>
<td>String</td>
<td>Unique order product identifier.</td>
<td>30.0</td>
</tr>
</tbody>
</table>

Custom Objects

<table>
<thead>
<tr>
<th>Property</th>
<th>Type</th>
<th>Description</th>
<th>Since Version</th>
</tr>
</thead>
<tbody>
<tr>
<td>attributes</td>
<td>Attributes</td>
<td>Type and URL of the custom object.</td>
<td>30.0</td>
</tr>
<tr>
<td>Id</td>
<td>String</td>
<td>Unique custom object identifier.</td>
<td>30.0</td>
</tr>
</tbody>
</table>

Request parameters

- You can use parameters for all standard and custom fields on contracts, orders, order products, and any custom objects directly related to these objects.
- The parameters must be fully qualified. For example: `objectname.relationshipname.fieldname=value`
  - Object name must be all lower-case.
  - Relationship name must match what is defined on the object and is case-sensitive.
  - Field name isn’t case sensitive.
  - If the value is a string, it must be encased in single quotation marks. If the value is a number, it must not be encased. If the value is a date, it should be in the `YYYY-MM-DD` format.
- You can use multiple parameter fields, separated by `&`, to make more detailed filters. For example:

```
/services/data/v30.0/commerce/sale/{contractId}?contract.status='Activated'
```

The following aren’t supported.
Arrays of values. For example: `order.orderItems.effectiveDate=2013-01-01,2013-01-02`.
- Operators: `>`, `>=`, `<` and `<=`
- The OR condition

<table>
<thead>
<tr>
<th>Parameters</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>order</td>
<td>The object name of the record being filtered. In this resource, this is always <code>order</code>.</td>
</tr>
<tr>
<td>orderItems</td>
<td>The relationship name of the field that the order's data will be filtered by. In this resource, this is always <code>orderItems</code>.</td>
</tr>
<tr>
<td>field name</td>
<td>The field whose value to filter by. For example, if you want to only retrieve order products with a certain start date, the field name is <code>effectiveDate</code>.</td>
</tr>
<tr>
<td>value</td>
<td>The value to filter by. For example, if you want to only retrieve order products that started on January 1, 2013, the value is <code>2013-01-01</code>.</td>
</tr>
</tbody>
</table>

### Response body

<table>
<thead>
<tr>
<th>Property</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>totalSize</td>
<td>Number</td>
<td>Total number of records listed.</td>
</tr>
<tr>
<td>records</td>
<td>Records[]</td>
<td>Attributes and IDs of the new records.</td>
</tr>
</tbody>
</table>

### Records

<table>
<thead>
<tr>
<th>Property</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>attributes</td>
<td>Attributes</td>
<td>Type and URL for the record.</td>
</tr>
<tr>
<td>Id</td>
<td>String</td>
<td>Unique record identifier.</td>
</tr>
</tbody>
</table>

### Attributes

<table>
<thead>
<tr>
<th>Property</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>type</td>
<td>String</td>
<td>Format of the resource.</td>
</tr>
<tr>
<td>url</td>
<td>URI</td>
<td>Resource URL.</td>
</tr>
</tbody>
</table>

### Examples

- Add Order Products to an Existing Order on page 23
- Get Details About an Order on page 26
Resource Reference

- Filter Details About an Order on page 28
INDEX

A
Add order products
on existing order 23
Authentication
Additional resources 16
OAuth 2–3, 9, 12, 14
OAuth endpoints 3
Remote access applications 2

G
Get contract data
filtered 27
unfiltered 24
Get order data
filtered 28
unfiltered 26

I
Introduction 1

L
Limitations 1
Limits, general 1

O
OAuth
Additional resources 16
Refresh token 14
User-agent OAuth flow 9
Username-password OAuth flow 12
Web server OAuth flow 3

P
Place order with order products
on existing account 19
on existing contract 20
on new contract 17
standalone 19

R
Reference 30
Requirements 1
Resource
GET data for an order 38
GET filtered data for an order 38
PATCH custom object, on existing order 38
PATCH custom object, on new order product 38
PATCH existing order 38
PATCH order products, on existing order 38
Resources
examples 17
GET data for a contract 33
GET filtered data for a contract 33
PATCH custom object, on existing contract 33
PATCH custom object, on new order 33
PATCH existing contract 33
PATCH orders and order products, on existing contract 33
POST an order 36
POST contract 30
POST custom object records, on contract-based orders 30
POST custom object records, on new contract 30
POST custom object records, on new order 36
POST custom object records, on new order products 36
POST order products, on new contract-based orders 30
POST order products, on new order 36
POST orders, on new contract 30