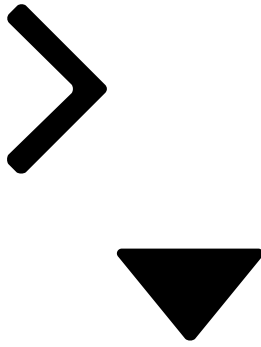


## 1. [Lightning Platform API Basics](#)



## 2. [Use SOAP API](#)

# Use SOAP API

## Learning Objectives

After completing this unit, you'll be able to:

- Generate a WSDL file for your org.
- Use SoapUI to create a SOAP project from the WSDL file.
- Log in to your Trailhead Playground using SOAP API.
- Create an account using SOAP API.

## Enterprise and Partner WSDLs

If you've sailed the straits of another SOAP-based API, you know that the Web Services Description Language (WSDL) file is basically your map to understanding how to use the API. It contains the bindings, protocols, and objects to make API calls.

Salesforce provides two SOAP API WSDLs for two different use cases. The enterprise WSDL is optimized for a single Salesforce org. It's strongly typed, and it reflects your org's specific configuration, meaning that two enterprise WSDL files generated from two different orgs contain different information.

The partner WSDL is optimized for use with many Salesforce orgs. It's loosely typed, and it doesn't change based on an org's specific configuration.

Typically, if you're writing an integration for a single Salesforce org, use the enterprise WSDL. For several orgs, use the partner WSDL.

For this unit, we're using the enterprise WSDL to explore SOAP API. The first step is to generate a WSDL file for your org. In your Trailhead Playground, from Setup, enter **API** in the Quick Find box, then select **API**. On the API WSDL page, click **Generate Enterprise WSDL**.

### API WSDL

Help for this Page ?

Salesforce's WSDL allows you to easily integrate salesforce.com with your applications, and to build new applications that work with salesforce.com. To get started, download a WSDL file to a place accessible to your development environment. For complete documentation, sample code, and developer community, visit <http://developer.salesforce.com/>

---

#### WSDL and Client Certificates

**Enterprise WSDL**  
A strongly typed WSDL for customers who want to build an integration with their salesforce.com organization only.  
[Generate Enterprise WSDL](#)

**Partner WSDL**  
A loosely typed WSDL for customers, partners, and ISVs who are building client applications for multiple organizations. It can be used to access data within any organization.  
[Generate Partner WSDL](#)

**Apex WSDL**  
Click on the link below to download an Apex programming WSDL.  
[Generate Apex WSDL](#)

On the Generate Enterprise WSDL page, click **Generate**. When the WSDL is generated, right-click on the page and save the WSDL file somewhere on your computer. We'll be using it shortly.

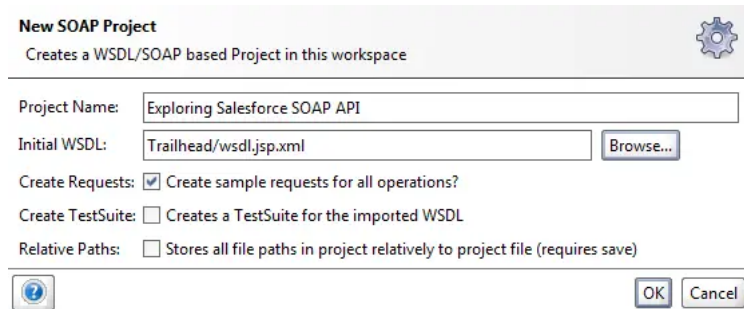
Here's a tip for working with the enterprise WSDL. The enterprise WSDL reflects your org's specific configuration, so whenever you make a metadata change to your org, regenerate the WSDL file. This way, the WSDL file doesn't fall out of sync with your org's configuration.

## Create a SOAP Project with SoapUI

Now that we have our WSDL file, we need a way to extract the information to start making SOAP API requests. In web parlance, this process is called consuming the WSDL. Much like a kraken consumes a ship full of hapless sailors, tools such as the Web Services Connector (WSC) consume the WSDL file. The tools then create classes that enable you to make requests with SOAP API using common programming languages.

For this unit, we're using a third-party tool called SoapUI to consume our enterprise WSDL file. SoapUI is a free and open-source app for testing web services. To get started, download and install SoapUI OpenSource from the [SoapUI website](#). Install only the SoapUI component.

After you get SoapUI installed and launched, from the File menu, select **New SOAP Project**. For the project name, enter `Exploring Salesforce SOAP API`. For the initial WSDL, browse to where you saved the enterprise WSDL file and select it. Don't change any other options. Your SoapUI window looks something like this.



**New SOAP Project**  
Creates a WSDL/SOAP based Project in this workspace

Project Name:

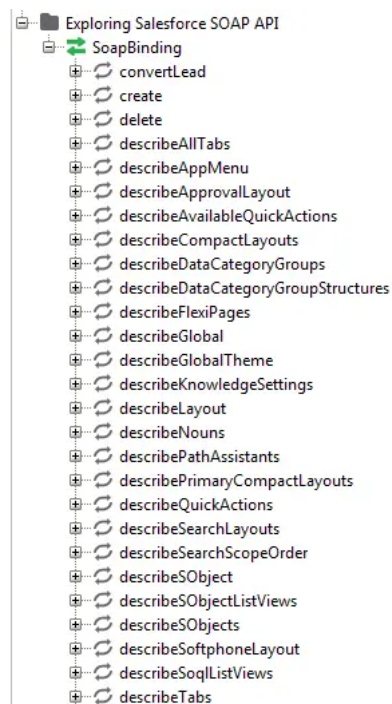
Initial WSDL:

Create Requests: ☒ Create sample requests for all operations?

Create TestSuite: ☐ Creates a TestSuite for the imported WSDL

Relative Paths: ☐ Stores all file paths in project relatively to project file (requires save)

Click **OK**. After a few seconds of processing, an Exploring Salesforce SOAP API folder appears in the navigator pane on the left side of the screen. Underneath it is an entry called SoapBinding that contains several operations.



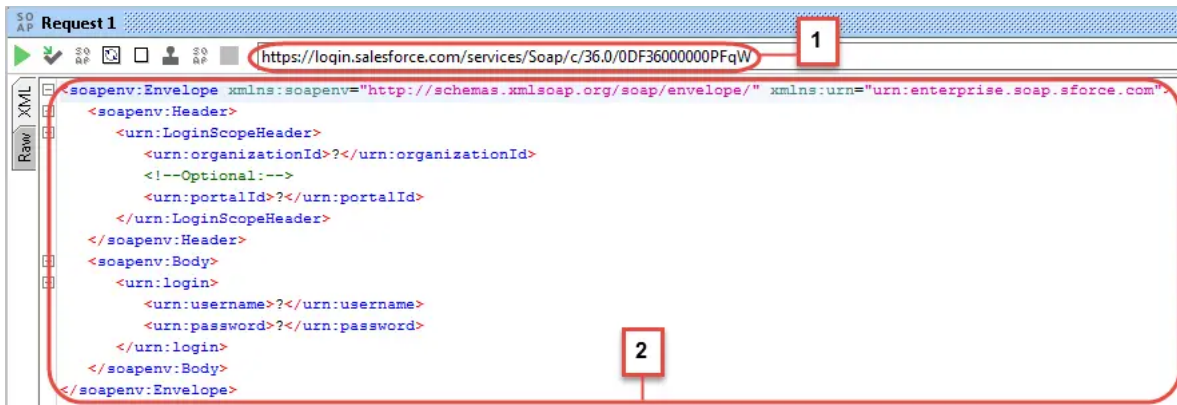
So what are we looking at here? Each operation corresponds to a SOAP API request we can make. The properties of each operation are pulled from information in the WSDL file. Each operation also contains a sample XML request that includes the operation's HTTPS endpoint and a prepopulated SOAP message.

One more thing—Salesforce requires all connections to use TLS 1.1 or higher. If you're using SoapUI with Java 7, TLS 1.1 isn't enabled by default. If you try to connect to Salesforce with an old version of TLS, you'll get an error message. The good news is that this is a pretty simple fix. Check out this handy-dandy [blog post](#) for more information.

Now we're all set to make SOAP API requests. Full sail ahead, cap'n!

## Log In to Your Trailhead Playground

In SoapUI, scroll down to the `login` operation. Expand it, and then double-click **Request 1**. A sample SOAP login request appears.



Here's a quick breakdown of the endpoint URI (1).

- `https://` —Specifies secure HTTP.
- `login.salesforce.com` —Top-level domain for a login request.
- `/services/Soap` —Specifies that we're making a SOAP API request.
- `/c` —Specifies that we're using the enterprise WSDL. Use `/u` for the partner WSDL.
- `/36.0` —API version number. The `v` prefix is missing because some APIs include it before the version number, and some don't. This behavior is just a quirk of Salesforce APIs.
- `/0DF3600000LHZw` —Package version number.

We aren't using managed packages for this example, so we can remove the package version number from the end of the URI. Go ahead and do that now.

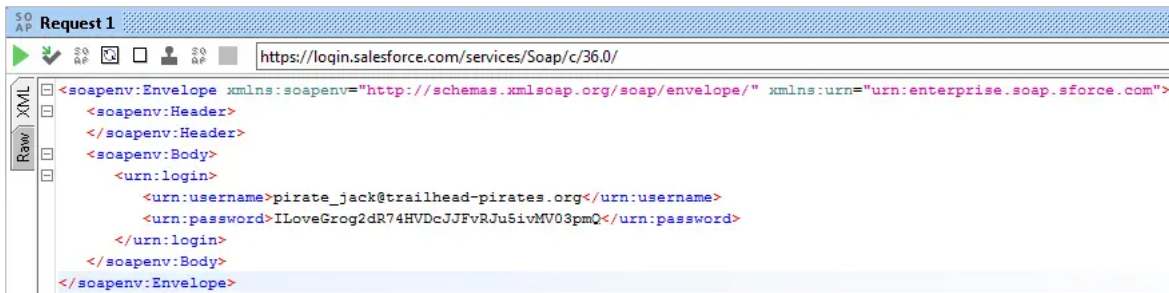
The SOAP message (2) contains everything we expect to find in a SOAP message: an envelope, a header, and a body.

The properties inside the `LoginScopeHeader` element concern the authentication of Self-Service and Customer Portal users. Because we don't have to worry about these values for our purposes, delete the entire `<urn:LoginScopeHeader>` element (everything from `<urn:LoginScopeHeader>` to `</urn:LoginScopeHeader>`). Highlight the text in the window, and press the Delete key.

Next, look at the `<urn:login>` element in the message body. This element is the bulk of the login request. It's where we provide our user credentials. Replace the `?`s with your username and password for your Trailhead Playground. This will require a password reset. To reset the password for your Trailhead Playground, follow [these instructions](#).

Since you're making an API request from an IP address unknown to Salesforce, you need to append your security token to the end of your password. For example, if your password is `mypassword` and your security token is `XXXXXXXXXX`, enter `mypasswordXXXXXXXXXX` inside the `<urn:password>` element. Your security token is emailed to you whenever you reset your password.

Your SOAP message looks something like this.



Click the play button (green triangle) in the upper left of the request window. This button sends the request, casting your SOAP message-in-a-bottle into the sea. Here's what it looks like when we expand the response.

```
<soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/" xmlns="urn:enterprise:soap:sforce.com"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
  <soapenv:Body>
    <loginResponse>
      <result>
        <metadataServerUrl>https://na30.salesforce.com/services/Soap/m/36.0/00D36000000wCdk</metadataServerUrl>
        <passwordExpired>false</passwordExpired>
        <sandbox>false</sandbox>
        <serverUrl>https://na30.salesforce.com/services/Soap/c/36.0/00D36000000wCdk</serverUrl>
        <sessionId>00D36000000wCdk!AQEAQK23h7Ak_SyGtu_WrzC1a2KTGnZacSOWiR4jc8Mp2Jt7_f4t8XkKJ.g64W1P7_JfweOHn.Ak.SwUEkQR0XGcKAXKq4gtU</sessionId>
        <userId>00536000002BU42AAG</userId>
        <userInfo>
          <accessibilityMode>false</accessibilityMode>
          <currencySymbol>$</currencySymbol>
          <orgAttachmentFileSizeLimit>5242880</orgAttachmentFileSizeLimit>
          <orgDefaultCurrencyIsoCode>USD</orgDefaultCurrencyIsoCode>
          <orgDisallowHtmlAttachments>false</orgDisallowHtmlAttachments>
          <orgHasPersonAccounts>false</orgHasPersonAccounts>
          <organizationId>00D36000000wCdkEEE</organizationId>
          <organizationMultiCurrency>false</organizationMultiCurrency>
          <organizationName>Pirates, Inc.</organizationName>
          <profileId>00e36000001JVJdAAO</profileId>
          <roleId xsi:nil="true"/>
          <sessionSecondsValid>7200</sessionSecondsValid>
          <userDefaultCurrencyIsoCode xsi:nil="true"/>
          <userEmail>pirate_jack@trailhead-pirates.org</userEmail>
          <userFullName>Pirate Jack</userFullName>
          <userId>00536000002BU42AAG</userId>
          <userLanguage>en_US</userLanguage>
          <userLocale>en_US</userLocale>
          <userName>pirate_jack@trailhead-pirates.org</userName>
          <userTimeZone>America/Los_Angeles</userTimeZone>
          <userType>Standard</userType>
          <userUiSkin>Theme3</userUiSkin>
        </userInfo>
      </result>
    </loginResponse>
  </soapenv:Body>
</soapenv:Envelope>
```

Congratulations, captain. You successfully logged in. The response contains a bunch of information about your org and user. Most importantly, it contains your org's instance (or custom domain, if you're using My Domain) and a session ID that we'll use to make future requests, highlighted here.

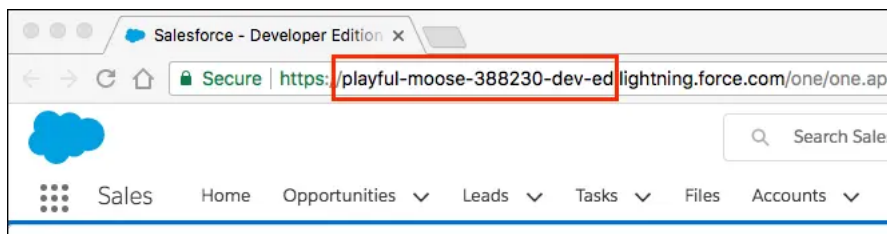
```
<soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/" xmlns="urn:enterprise:soap:sforce.com"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
  <soapenv:Body>
    <loginResponse>
      <result>
        <metadataServerUrl>https://na30.salesforce.com/services/Soap/m/36.0/00D36000000wCdk</metadataServerUrl>
        <passwordExpired>false</passwordExpired>
        <sandbox>false</sandbox>
        <serverUrl>https://na30.salesforce.com/services/Soap/c/36.0/00D36000000wCdk</serverUrl>
        <sessionId>00D36000000wCdk!AQEAQK23h7Ak_SyGtu_WrzC1a2KTGnZacSOWiR4jc8Mp2Jt7_f4t8XkKJ.g64W1P7_JfweOHn.Ak.SwUEkQR0XGcKAXKq4gtU</sessionId>
        <userId>00536000002BU42AAG</userId>
```

Copy the instance and session ID into a text file. We use them in a minute.

Because your org's instance is likely to change, don't hard-code references to the instance when you start building integrations! Instead, use the Salesforce feature My Domain to configure a custom domain. A custom domain not only eliminates the headache of changing instance names, you can use it to highlight your brand, make your org more secure, and personalize your login page.

## My Domain Is Already On in Your Trailhead Playground

Do not attempt to turn on My Domain, or change its settings, in your Trailhead Playground. By default, My Domain is already active in every Trailhead Playground.



In your production org, My Domain lets you create a subdomain unique to your organization. With My Domain, you replace the instance URL that Salesforce assigns you, such as <https://na17.lightning.force.com>, with your chosen subdomain, for example, <https://mydomainname.lightning.force.com>.

My Domain is required to create custom Lightning components and set up single sign-on (SSO) in an org. To learn more about My Domain, check out this [knowledge article](#). To learn how to activate it in your production org, see the [User Authentication](#) module.

## Create an Account



Just like we did with REST API, let's create an account with SOAP API. In the navigation pane on the left side of the screen, find the `create` operation. Expand it, and double-click **Request 1**.

Because creating a record is more complicated than logging in, the `create()` SOAP message includes many more elements. Most of the elements are contained in the request header, and most of them are optional. To simplify things, we'll delete most of the header info, but remember that the options provided by these headers are available when you create a record. For information on what each header does, check out the SOAP Headers topic in the SOAP API Developer Guide.

One header we don't want to delete, though, is `SessionHeader`. It's going to contain the session ID we got from the `login()` response. Go ahead and delete the other headers, from `<urn:EmailHeader>` to `<urn:AssignmentRuleHeader>`. Your message looks like this.

```
<soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/" xmlns:urn="urn:enterprise.soap.sforce.com"
  xmlns:urn1="urn:object.enterprise.soap.sforce.com">
  <soapenv:Header>
    <urn:SessionHeader>
      <urn:sessionId?></urn:sessionId>
    </urn:SessionHeader>
  </soapenv:Header>
  <soapenv:Body>
    <urn:create>
      <!--Zero or more repetitions:-->
      <urn:sObjects>
        <!--Zero or more repetitions:-->
        <urn1:fieldsToNull?></urn1:fieldsToNull>
        <urn1:Id?></urn1:Id>
      </urn:sObjects>
    </urn:create>
  </soapenv:Body>
</soapenv:Envelope>
```

Get the session ID you copied to a text file and paste it inside the `<urn:sessionId>` tag, replacing the `?`.

We have a few more adjustments to make to the message body. First, we specify that we're creating an account. Change the text in the `<urn:sObjects>` tag to look like this: `<urn:sObjects xsi:type="urn1:Account" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">`. This adjustment specifies the correct record type using the XML instance schema declaration.

We also want to give the account a name. Add `<Name>Sample SOAP Account</Name>` inside the `sObjects` element. Delete the `fieldsToNull` and `Id` elements, too. Now your message looks something like this.

```
<soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/" xmlns:urn="urn:enterprise.soap.sforce.com"
  xmlns:urn1="urn:object.enterprise.soap.sforce.com">
  <soapenv:Header>
    <urn:SessionHeader>
      <urn:sessionId>00D36000000wCdk!AQEAQK23h7Ak_SyGTu_WrzCia2KTGnZacSOWiR4jc8Mp2Jt7_f4t8XkKJ.g64W1P7_JfweOHn.Ak.SwUEkQR0XGcKAXKq4gtU</urn:sessionId>
    </urn:SessionHeader>
  </soapenv:Header>
  <soapenv:Body>
    <urn:create>
      <!--Zero or more repetitions:-->
      <urn:sObjects xsi:type="urn1:Account" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
        <!--Zero or more repetitions:-->
        <Name>Sample SOAP Account</Name>
      </urn:sObjects>
    </urn:create>
  </soapenv:Body>
</soapenv:Envelope>
```

One last thing to do before we make the request. Change the endpoint to specify your org's instance rather than `login`, and remove the package version from the end of the URI. The endpoint URI looks something like this: `https://na30.salesforce.com/services/Soap/c/36.0`.

We're ready to send the request. Click the green triangle again. It worked! Let's look at the response.

```
<soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/" xmlns:urn="urn:enterprise.soap.sforce.com">
  <soapenv:Header>
    <LimitInfoHeader>
      <limitInfo>
        <current>9</current>
        <limit>15000</limit>
        <type>API REQUESTS</type>
      </limitInfo>
    </LimitInfoHeader>
  </soapenv:Header>
  <soapenv:Body>
    <createResponse>
      <result>
        <id>00136000000L0s2QAAR</id>
        <success>true</success>
      </result>
    </createResponse>
  </soapenv:Body>
</soapenv:Envelope>
```

Notice the `LimitInfoHeader`. This header returns information about your API usage. In the above example, we've made 9 out of 15,000 allowed calls today.

In the response body, notice the `<result>` element. `<success>true</success>` indicates that the record was created successfully. `<id>` includes the ID of the record, which you can use in future requests.

We covered the basics of making requests with SOAP API. Of course, each operation has its own parameters and peculiarities. Make sure that you use the SOAP API Developer Guide as your map when you start writing integrations with SOAP API.

## Resources

[SOAP API Developer Guide](#)

[SOAP API Cheatsheet](#)

[SOAP API Sample WSDL Structures](#)

[SOAP API Web Services Connector \(WSC\)](#)

[SoapUI Homepage](#)



### Note

Remember, this module is meant for Salesforce Classic. When you launch your hands-on org, switch to Salesforce Classic to complete this challenge.

## Assessment Complete!

**+500 points**



Lightning Platform API Basics

100%

Progress: 100%

Retake this Challenge

[View more modules](#)