WEBSERVICER

User's Guide

Table of Contents

Introduction

This document is a detailed setup guide for Webservicer - a native Force.com App enabling you to update records with values provided by public external SOAP web services in three simple steps. Upload WSDL file (Web service description), map inputs and outputs of web service functions, add custom button to page layout and you are ready to make callouts. And best of all, no code is required.

Overview

Callouts are invoked for a specific record via custom button, possibly taking some of the fields as inputs to web service and updating the record with returned values with respect to defined mapping. Connections are possible for SOAP web services only.

Being built on the Force.com platform, this App inherits some platform governor limits, namely:

- Maximum cumulative timeout for all callouts (HTTP requests or Web services calls) in a transaction 120 seconds
- Maximum size of callout request or response is 6 MB
- Before any Apex callout can call an external site, that site must be registered in the Remote Site Settings page, or the call will fail. The platform, by default, prevents calls to unauthorized network addresses
- Being saved as plain text in Long text area field, WSDL file can contain up to 131,072 characters

Callouts are **not** counted against API usage limits.

Please note the App does not support WSDL documents with imports, nor does it currently support HTTP Headers.

We recommend that a web service uses SSL for transmitting data (check your WSDL file- look for "<soap:address location=… />". The location URL is URL being called, it should start with **https**.)

The App currently supports up to 3 different WSDL files and up to 5 updated fields per Object (Standard or Custom).

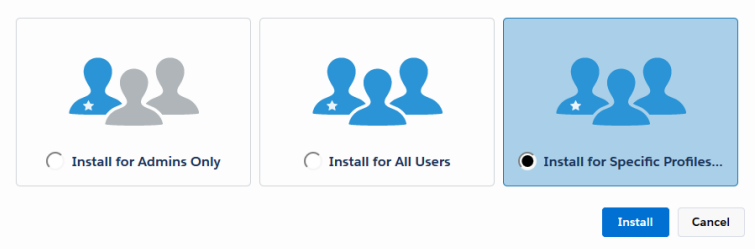
At the moment, App offers no form of authentication with remote servers.

Installation

Get the latest version of Webservicer via AppExchange (https://appexchange.salesforce.com/listingDetail?listingId=a0N3A00000EwYvGUAV).

Click **Get It Now**, then Install in production.

If installing in Enterprise, Unlimited, Performance, and Developer Editions, choose a security option- select **Install for Specific Profiles**.



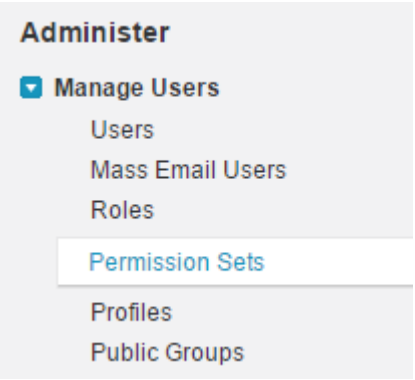
Scroll down to apply desired access level for each profile (Because permissions are not editable for standard profiles, this applies to custom profiles only). To modify access after installation, and for users with standard profiles use permission sets (see the next section). We included two custom profiles - "Webservicer Administrator" and "Webservicer User". The user profile does not allow users to change the App settings and does not allow access to the App itself, it only allows making callouts. Admin profile gives full access to all the features of the App.

Adding Permission Sets

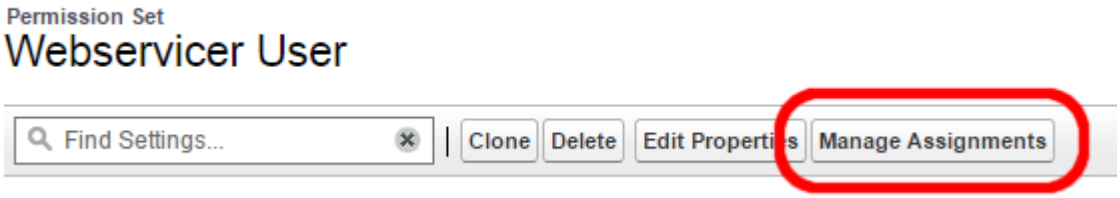
To set permissions for the App after installation and for user with standard profiles, assign permission sets to the users.

We prepared two permission sets, which should cover your needs. The "Webservicer User" permission set does not allow users to change settings and does not allow access to the App itself, only allows making callouts. "Webservicer Administrator" permission set gives full access to all the features of the App.

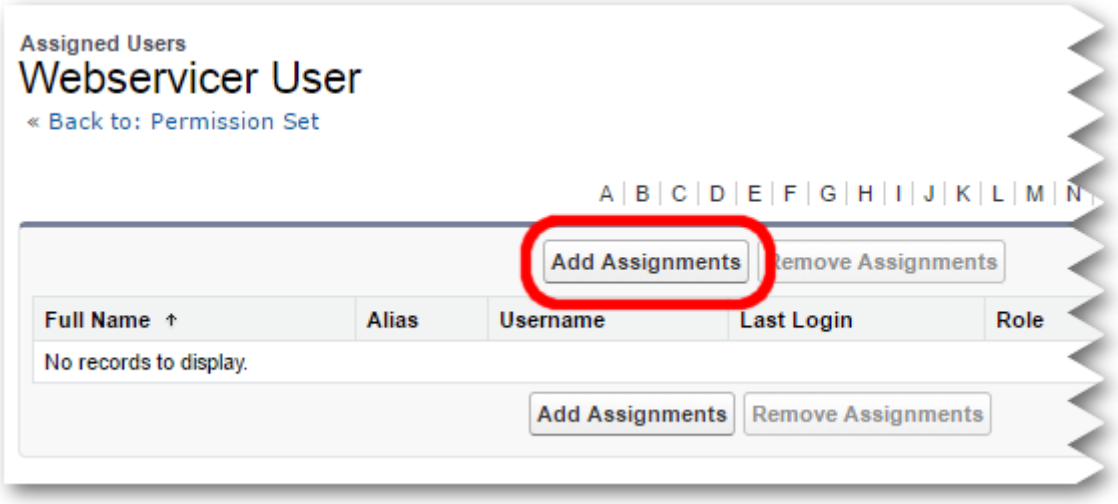
- In **Setup** under Administer, select **Manage Users** and select **Permission Sets**.



- Click on the permission set name and click **Manage Assignments** button.



- Then Click on **Add Assignments** button



Select desired users and click **Assign**.

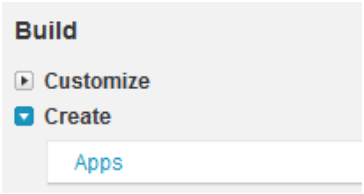
Repeat this for both permission sets.

This method grants all needed permissions except for assigning App to the user, since this setting cannot be packaged in managed package. Therefore you need to assign Webservicer App to admin users manually.

You can either assign the App to specific profiles or create permission set to assign the App to specific users.

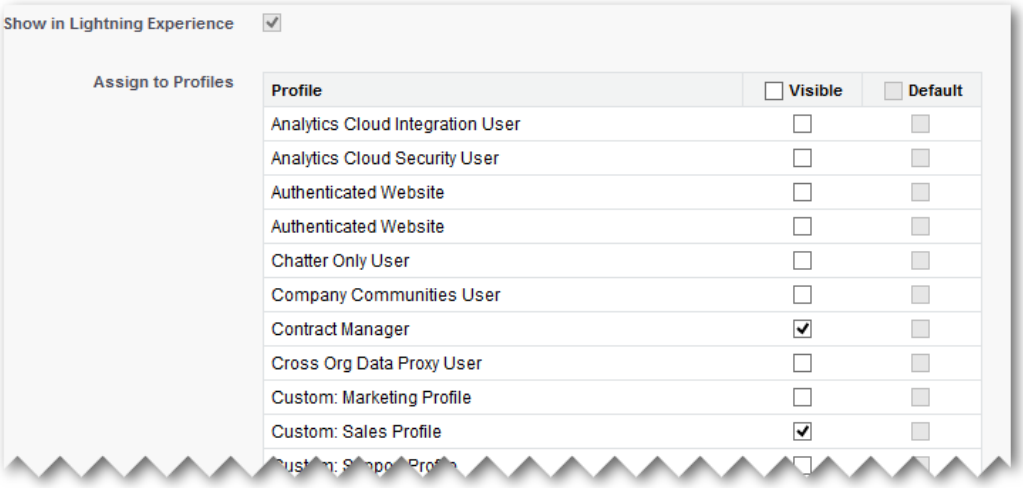
To assign App to profiles

In Setup under **Build** click **Create**, then select **Apps**.



Click **Edit** next to the Webservicer App.

In the **Assign to Profiles** section, select desired profiles. Click **Save**.

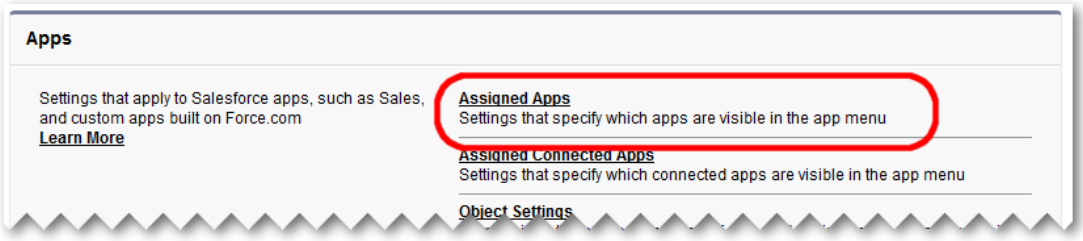


To assign App to users

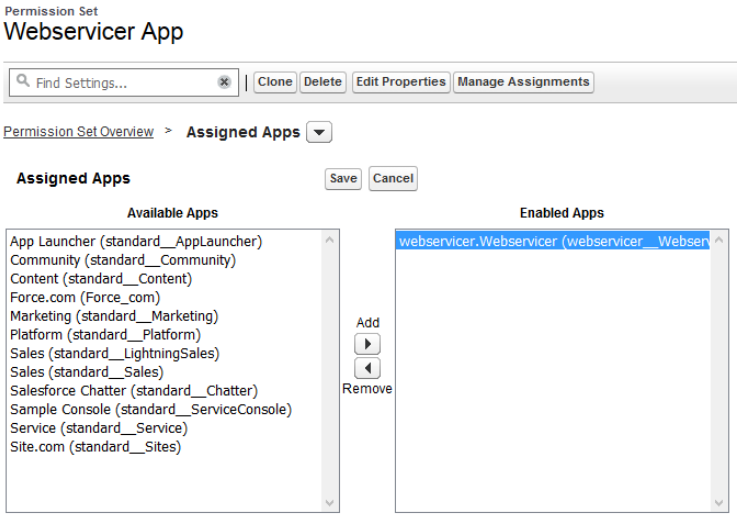
In Setup under **Administer** click **Manage Users** and select **Permission Sets**.

Click **New**, fill the fields and **Save** the permission set.

Select **Assigned Apps** and click **Edit**.



Add Webservicer to Enabled Apps and click **Save**.

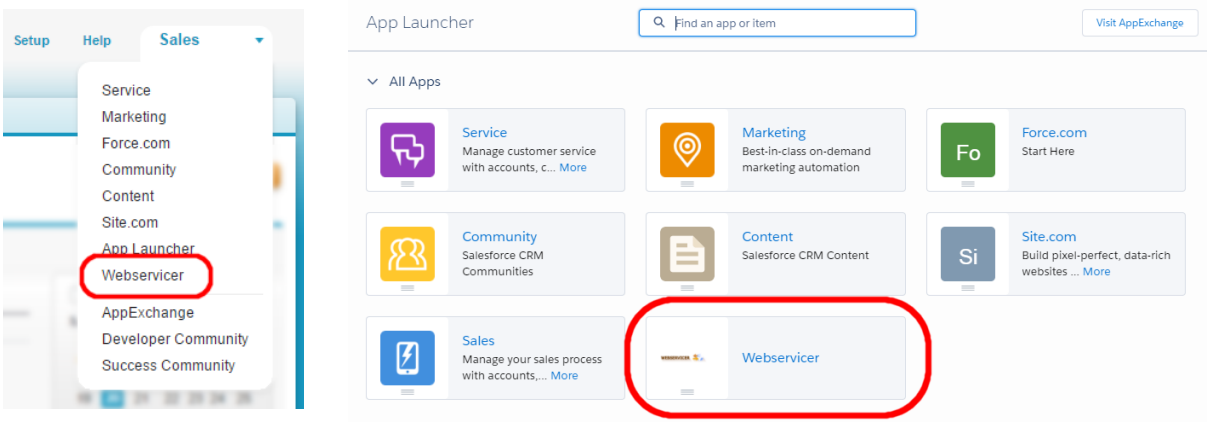


Assign this profile to desired users (see Adding Permission Sets section).

Setup

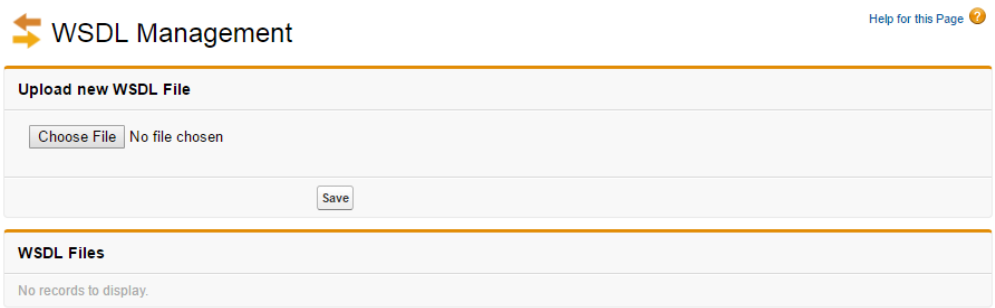
Importing WSDL files

First, open the Webservicer App.



Click on the "**WSDL**" Tab.

Click **Choose File**, navigate to a WSDL document on your local drive to upload (.xml, .asmx, .wdsl and .txt file types are supported) and click **Save**. App will validate the web service definition and will report any errors should they arise.

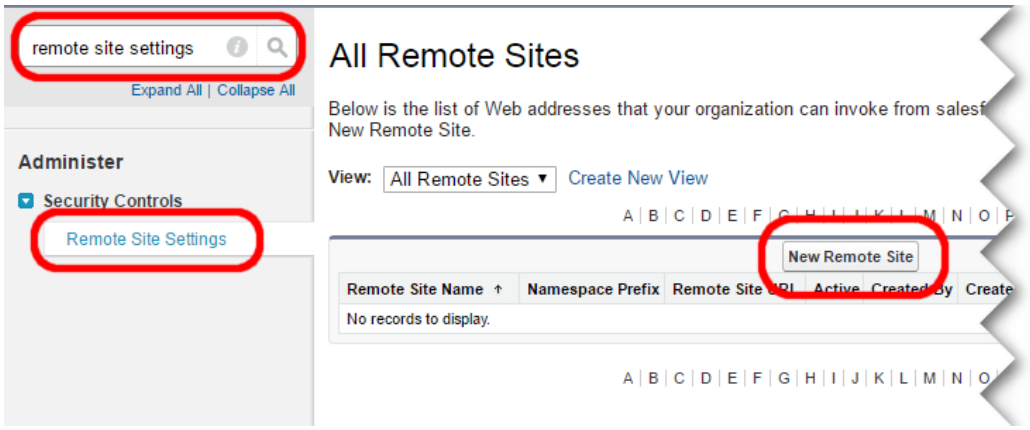


If the definition contains no errors it will appear in WSDL Files list after saving.

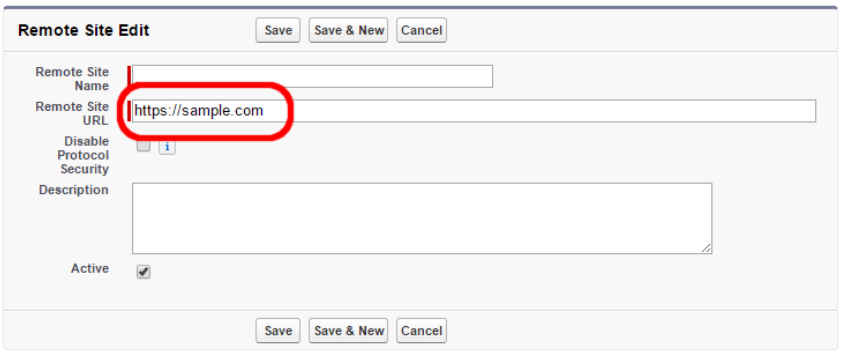
The WSDL Files table contains column indicating whether the web service endpoint is listed in Remote Site Settings.

If the checkbox is not checked, please add the URL from EndpointURL column to Remote Site Settings before proceeding. This step is necessary because any HttpRequest that calls an external site must be registered, or the call fails. You can find additional information in Salesforce Documentation.

In **Setup**, search for "remote site settings", under **Security Controls** click **Remote Site Settings** and click **New Remote Site**.



Paste EndpointURL address to Remote Site URL field. Set the name and click **Save**.



After making sure the web service endpoint is registered, you can now map remote web service functions to Standard or Custom Objects.

Testing web services

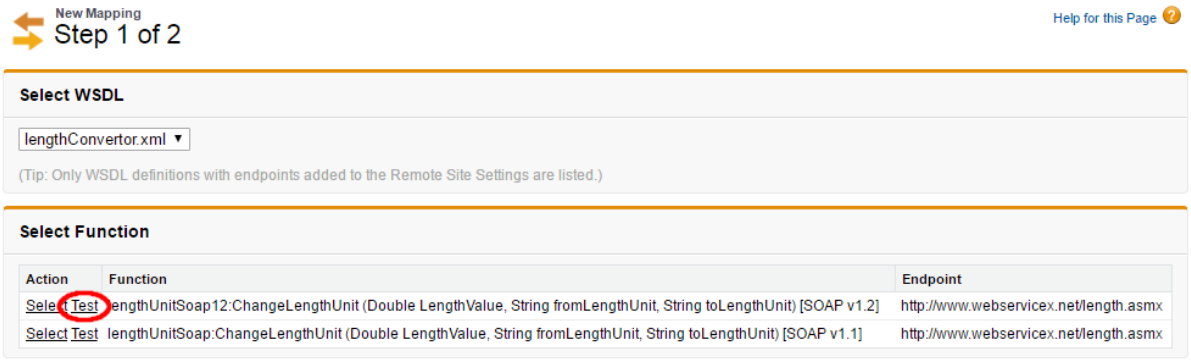
Before mapping web services to Objects in your Organization, you have an option to send HTTP Callouts to observe what the web service response is with given inputs, although you are expected to be familiar with given web service and expected input and output values already.

Start by selecting the "Map" Tab in Webservicer App.

Select WSDL definition from dropdown list of all uploaded files. If you cannot see your WSDL listed, make sure the endpoint URL is registered in Remote Site Settings (See previous section).

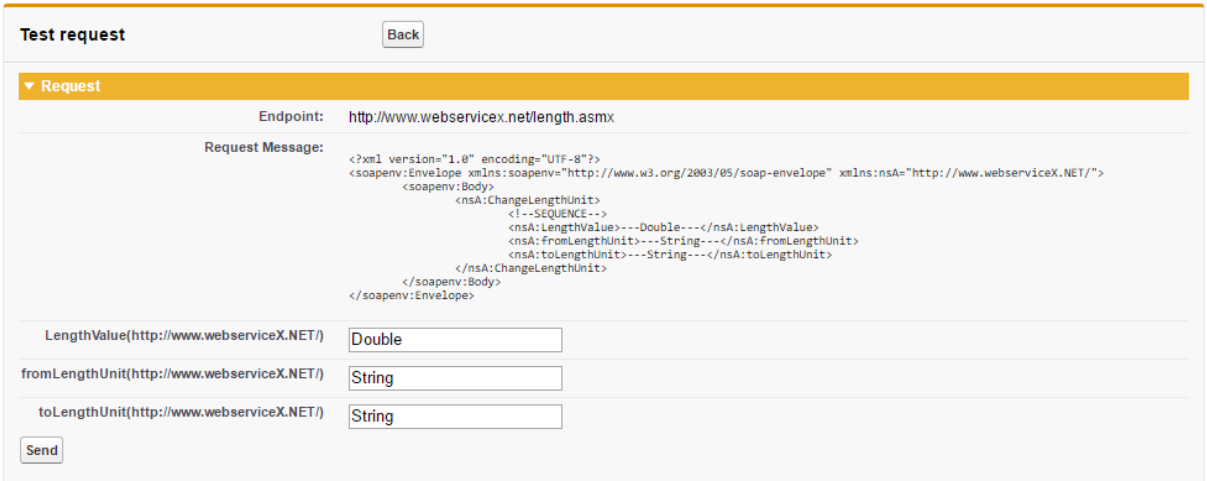
Click **Test** next to the web service function you want to test. You can distinguish functions by their name and input parameters.

(Following example has been made with the use of free public [web service by www.webservicex.net](http://www.webservicex.net))

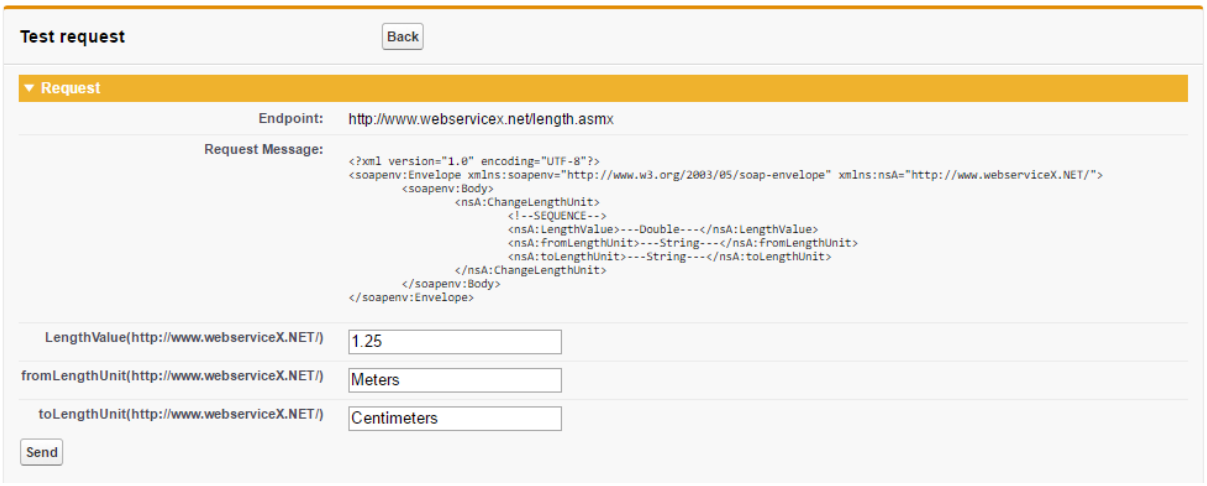


In **Request** section, you can see the endpoint to which request will be sent, sample SOAP message that will serve as body of request and Input fields for parameters that need to be filled in the message.

Input fields are pre-filled with expected parameter type.

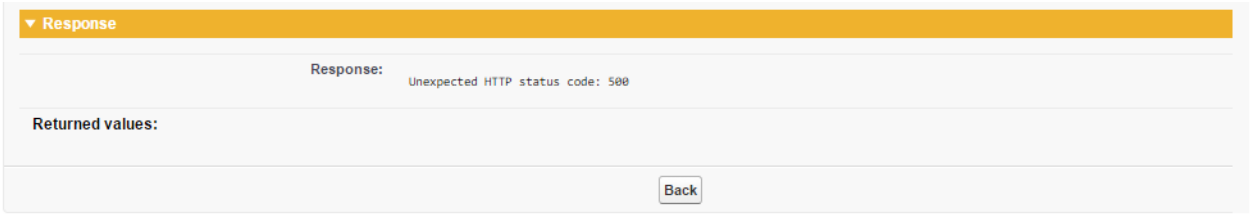


Fill input fields and click **Send**.



App then makes a callout to web service and if response is returned successfully it is displayed along with parsed return values in the **Response** section.

If any problem occurred errors are displayed.

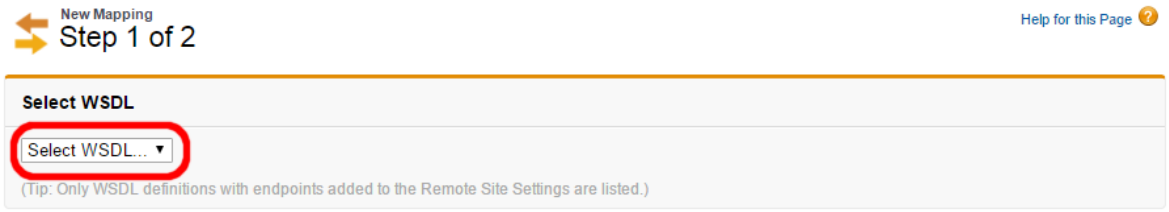


Mapping web services to objects

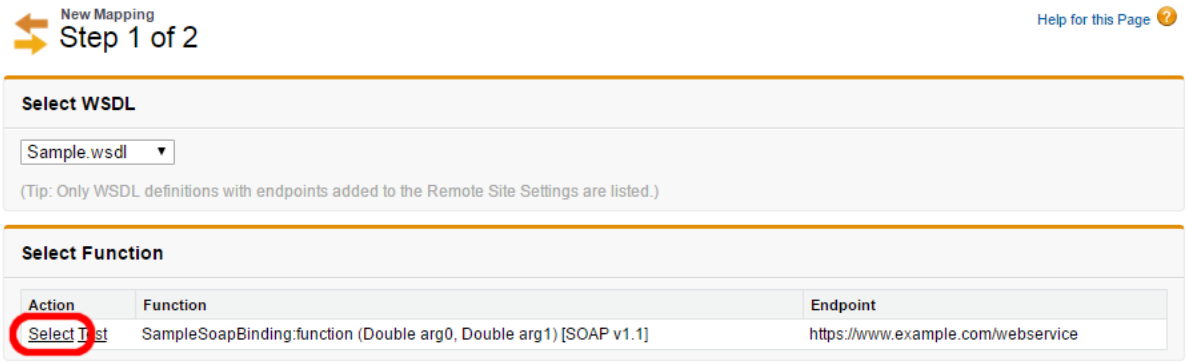
Click on "Map" Tab in Webservicer App.

Step 1

Select WSDL definition from dropdown list of all uploaded files. If you cannot see your WSDL listed, make sure the endpoint URL is registered in Remote Site Settings.



All defined remote functions of web service are listed. Select a web service function you want to use by clicking **Select** next to the function. You can distinguish functions by their name and input parameters.

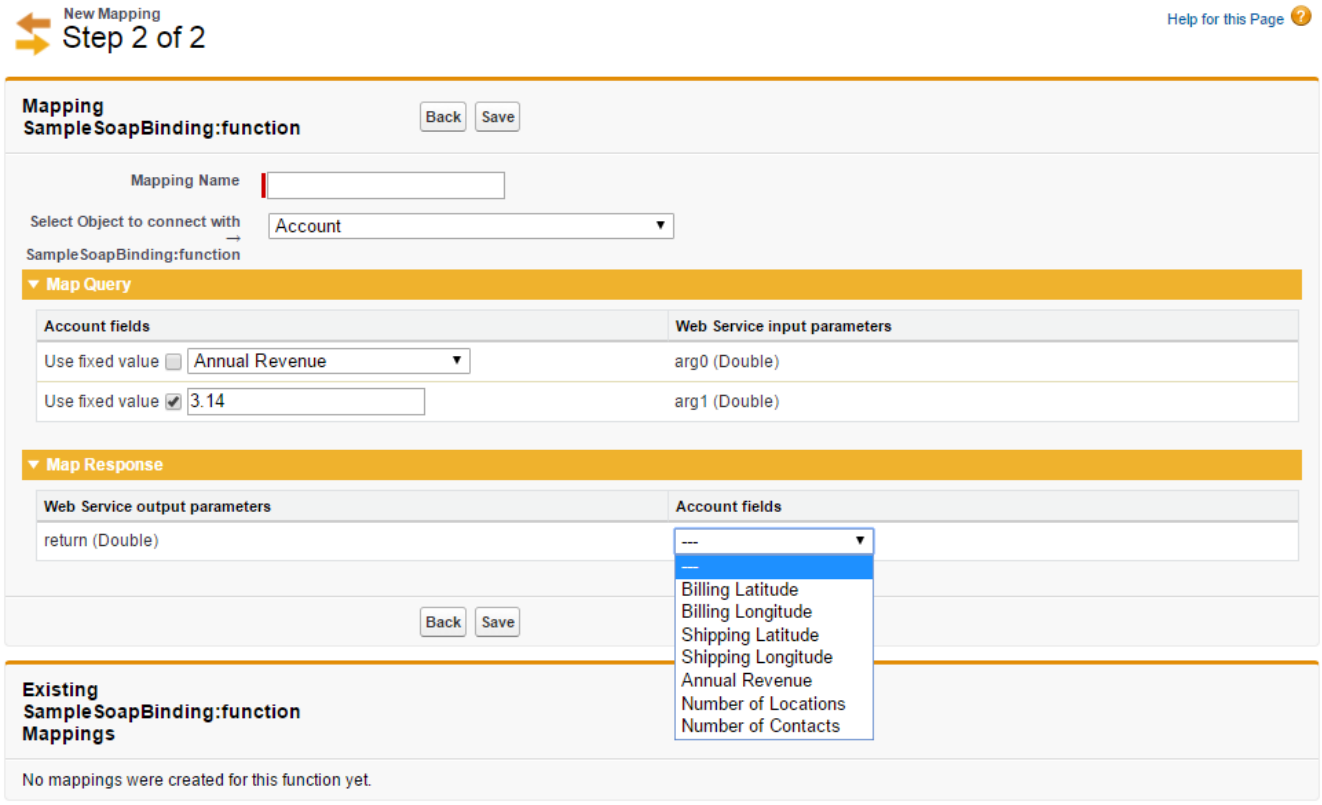


Step 2

- Name your mapping so you can easily identify it later.
- In "Select Object to connect with → ..." dropdown list, choose Object you wish to run web service on. Please note that not all listed Objects are suitable to be used with external web services. We leave this decision to your discretion, though we advise you to limit usage of this App to commonly used Standard Objects (such as Accounts, Opportunities, Contacts...) and Custom Objects.
- In Map Query section, you can select:
    - Object field - value of this field will be sent as input parameter to web service.
    - Manually typed fixed value to always send to web service independent of Object record.

  Expected type is denoted after web service input parameter name, however you can choose field of any type should you deem the field fit as input to web service regardless of type. All inputs must be defined.

- In Map Response section, select Object fields to which values returned from web service will be saved. These fields are filtered to compatible types. You don't have to use all returned values.



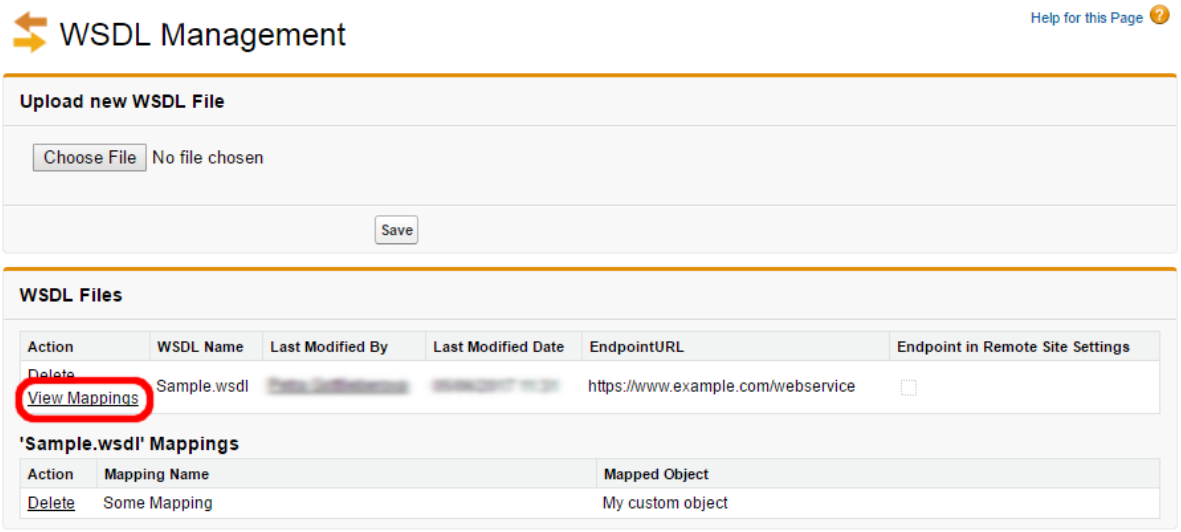
Click **Save**. If no errors occurred new mapping is created and is ready to be used.

Manage Mappings

To view or remove Mappings associated with a WSDL document:

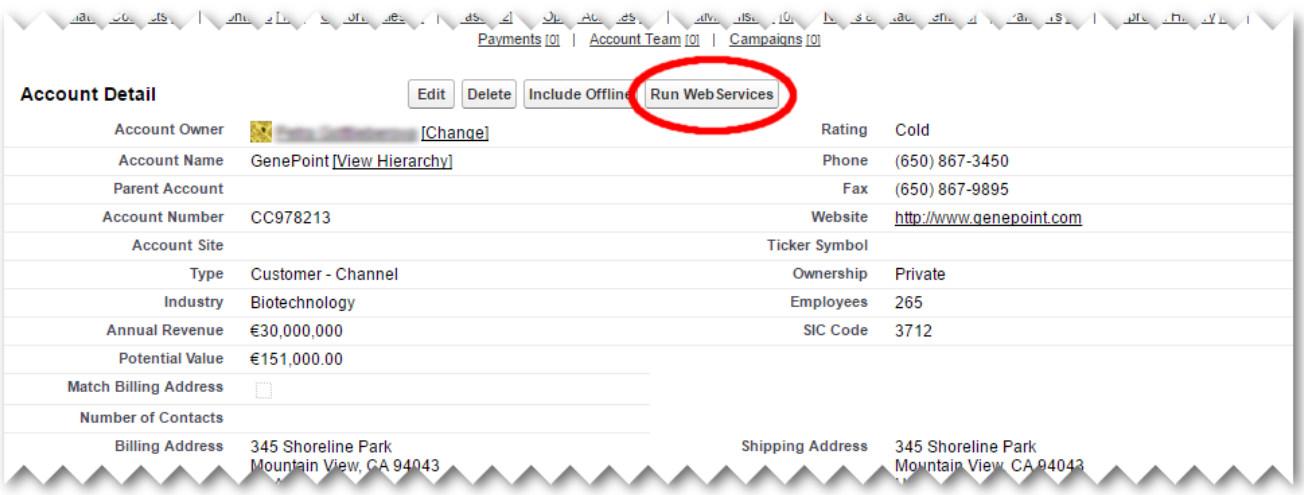
- Click "WSDL" Tab.

- Click **View Mappings** for desired WSDL file. List of mappings belonging to this web service is shown. Notice column "Mapped Object" with name of the Object utilizing this mapping.



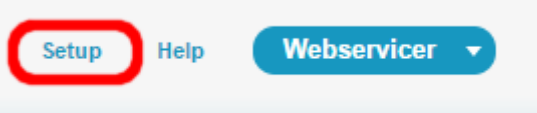
Preparing Page Layouts

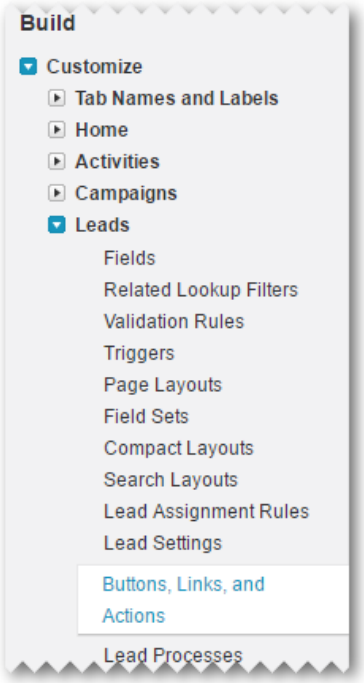
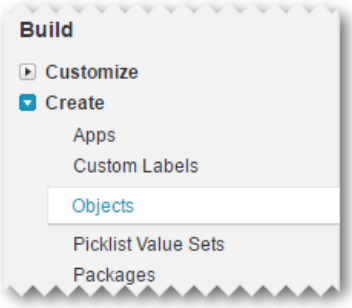
To make callouts to mapped web services, user needs to press button labeled **Run web services** from Object record page. For your convenience, we created this button for Account, Lead, Contact and Opportunity objects, however for other Standard Objects and Custom Objects you need to create this button yourself. You will also need to include it in Page Layouts manually.

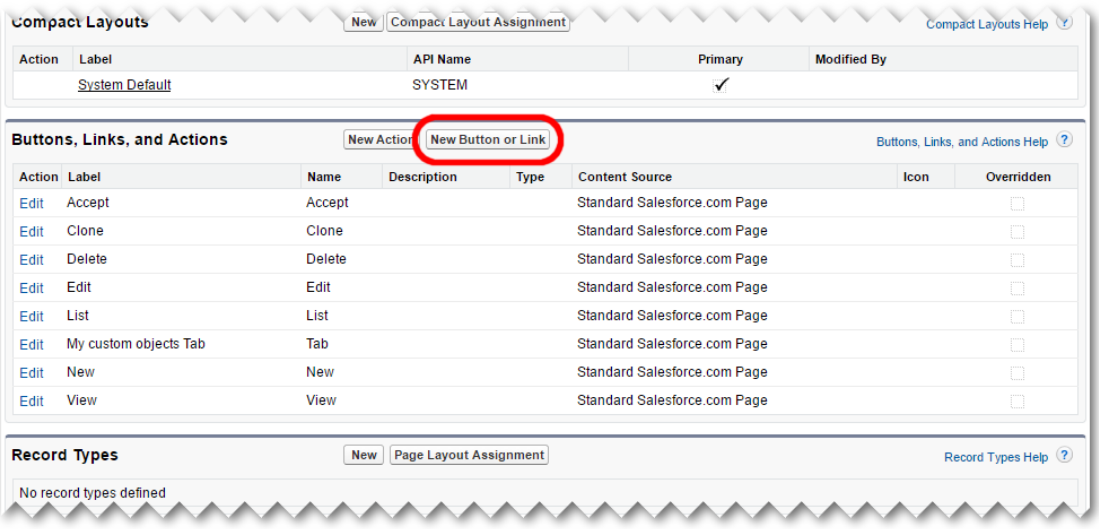


Creating Button

Go to **Setup**.

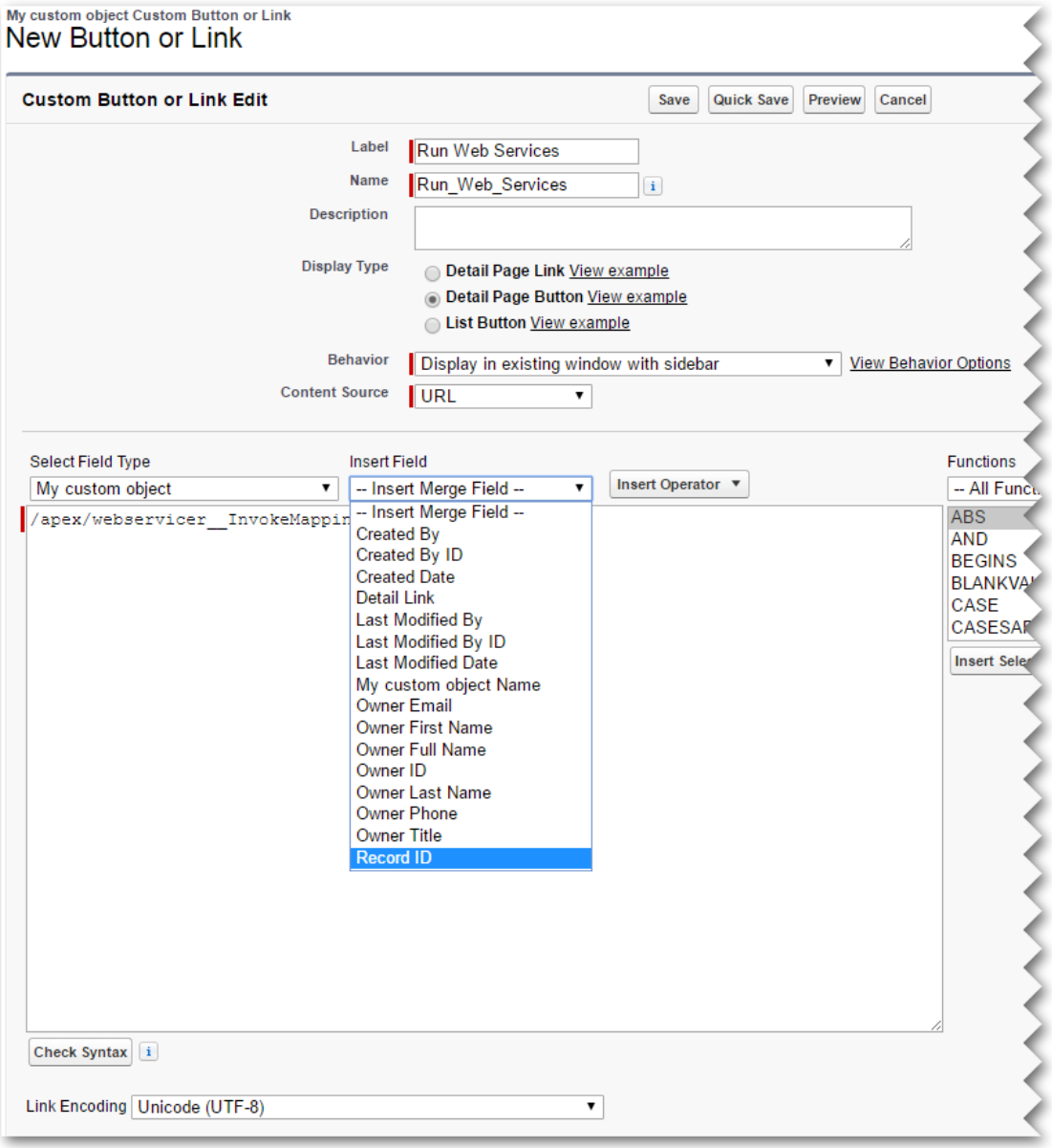


For Standard Objects	For Custom Objects
• Click **Customize** • Click Object name • Click **Buttons, Links, and Actions** • Click **New Button or Link**	• Click **Create** • Click **Objects** • Click on Object name in the list • Under **Buttons, Links, and Actions** section click **New Button or Link**
	



Fill the form with following:

- **Label**: Run Web Services
- **Name**: Run_Web_Services
- **Display Type**: Detail Page Button
- **Behavior**: Display in existing window with sidebar
- **Content Source**: URL
- Into the form field paste "**/apex/webservicer__InvokeMapping?id=**" (without quotation marks), then click Insert Field and select Object ID field so the link looks like this: /apex/webservicer__InvokeMapping?id={!Account.Id} (for Account)

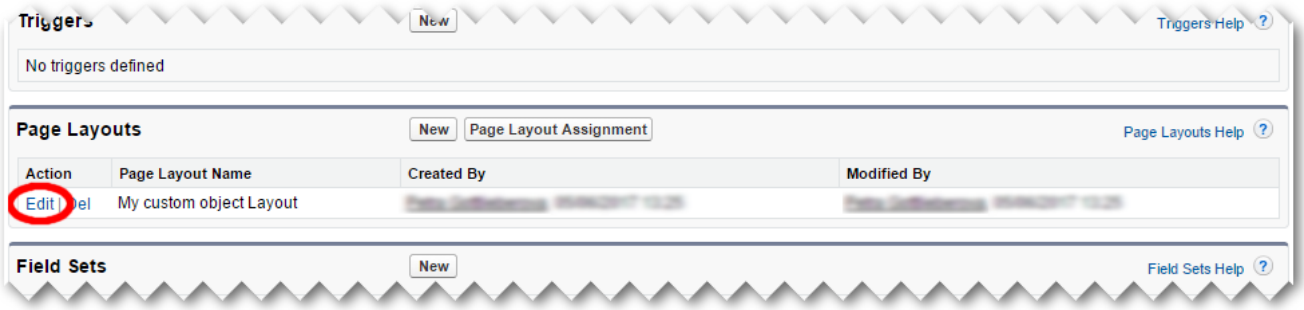


Click **Save**.

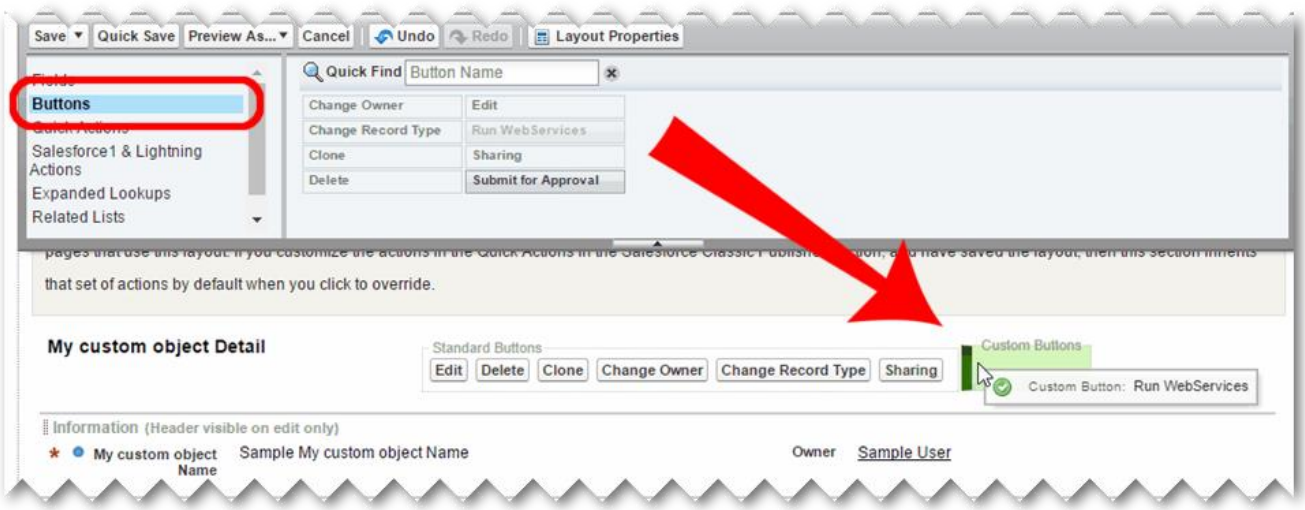
Adding the Button to Page Layout

Now you need to add the button you just created to Page Layout for the given Object.

In **Setup**, under **Page Layouts** section for Standard or Custom Object, click **Edit** next to the Layout where you want to have the button displayed.



On the Page layout edit page, from top menu choose **Buttons**. Drag and drop the new button (with the name you chose in the previous step) into the **Custom Buttons** section. Click **Save**.

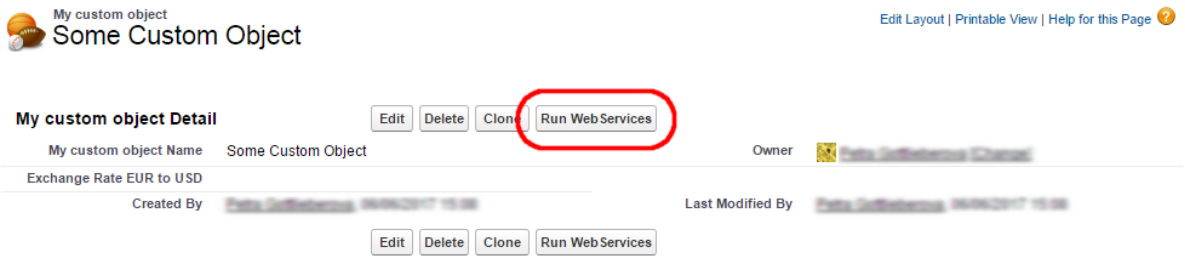


Repeat this for every layout you need to display the button on.

Usage

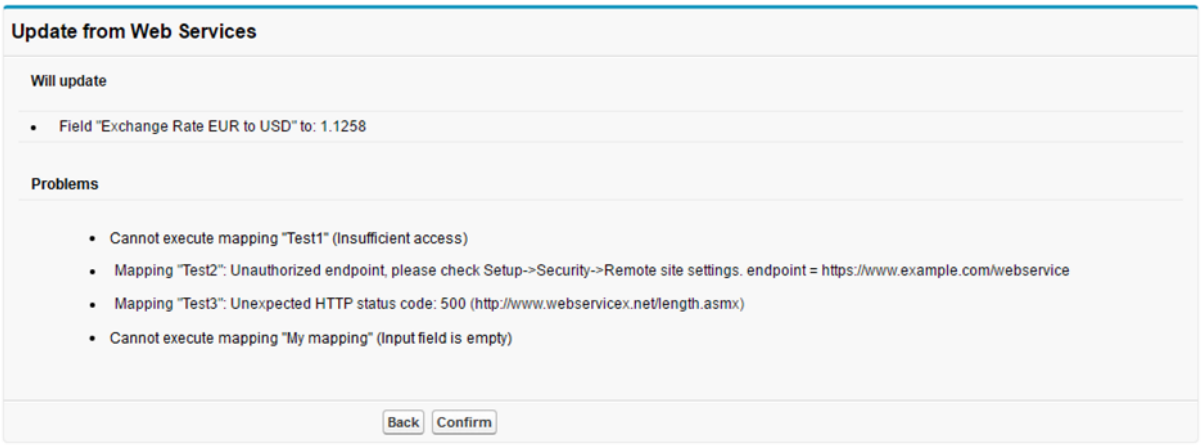
To update the record with values provided by web services, navigate to Detail Page of given record.

Click **Run Web Services** button.

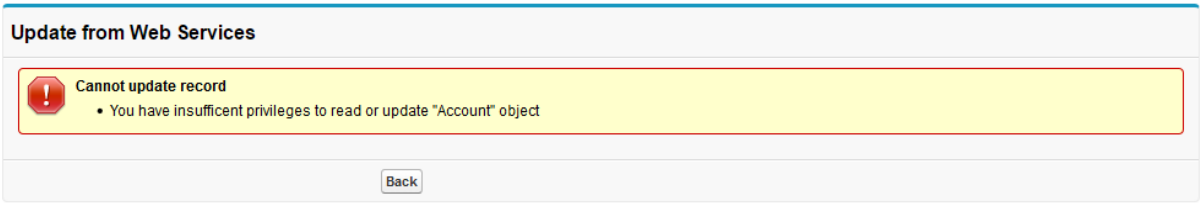


If user permissions allows it, all mapped web services for the Object are called and following information (if applicable) is displayed:

- List of fields that will be updated and their new values.
- List of problems that occurred (e.g. mapped return field is not updateable by current user or mapped request field is not readable by current user)

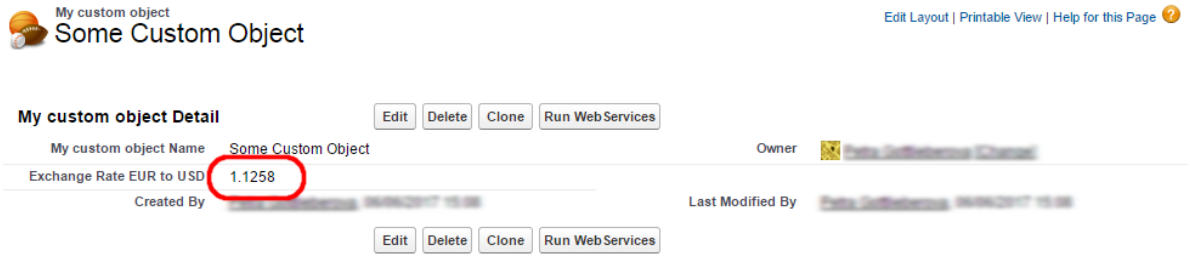


- Errors



If you wish to update record with new values, click **Confirm**. Otherwise click **Back**.

You will be taken back to the record Detail Page and, if confirmed, new values are saved.



Please note, that when using Lightning Experience, sometimes old values are still displayed on the object's detail page. Record is correctly saved to database and new values should display after browser refresh.

Support

For more information and support, feel free to contact us at:

office@anodius.com