INTEGRATION ARCHITECTURE RESOURCE GUIDE

Table of Contents

Copyright

Introduction

Who is the Salesforce Certified Integration Architecture Designer?

The candidate looking to obtain the Integration Architecture Designer Certification assesses the architecture environment and requirements and designs sound and scalable technical solutions on the Force.com platform that meet end-to-end integration requirements. The candidate has experience communicating solutions and design trade-offs to business stakeholders.

The experience and skills that the candidate should possess are outlined below:

- Has 5+ years of delivery experience.
- Provides experienced guidance on the appropriate choice of on-platform and off-platform technology.
- Understands integration capabilities and patterns, design trade-offs, and has the ability to communicate design choices.
- Has held a technical architect role on multiple complex deployments or has gained equivalent knowledge through participation and exposure to these types of projects [either with single or multiple projects].
- Has a thorough understanding of Web Services in general and SOAP and REST specifically; understands the basic workings of HTTP/S.
- Understands the different Force.com APIs and is able to design solutions using the appropriate API.
- Understands data migration considerations, design trade-offs, and common ETL tools.
- Has experience with common integration patterns used on the Force.com Platform.
- Understands patterns/mechanisms to secure integrations, such as TLS for HTTP.

Learn Materials

General Resources

Here are some comprehensive general resources that are a good starting place for your self-paced study.

Integrating with the Force.com Platform

This article provides an overview of the fundamental developer integration points available on the Force.com platform. After reading this article, you will be aware of approaches you could take, and you'll have enough pointers to more in-depth material to implement your integration.

Integration

A typical enterprise uses many applications, many or most of which are not designed to work with one another out of the box. Integrating separate but related apps helps organizations achieve greater levels of operational consistency, efficiency, and quality.

Integrating Applications with Force.com

One of the most frequent tasks Force.com developers undertake is integrating Force.com apps with existing applications.

UML 2 Sequence Diagrams: An Agile Introduction

UML sequence diagrams model the flow of logic within your system in a visual manner, enabling you to document and validate your logic, and are commonly used for both analysis and design purposes. Sequence diagrams are the most popular UML artifact for dynamic modeling, which focuses on identifying the behavior within your system.

Catalog of Patterns of Enterprise Application Architecture

These pages are a brief overview of each of the patterns in P of EAA. They aren't intended to stand alone, but are merely a quick aid for those familiar with them, and a handy link if you want to refer to one online.

Patterns in Enterprise Software

In recent years, there's been a small but useful growth in describing patterns for the development of enterprise systems. This page lists the most notable catalogs on these patterns and some thoughts on the broad interrelationships between them.

Please register in the Salesforce Success Community and join our Architect Success group here.

General Overview

The following pages will introduce you to various focuses within the Integration Architecture. You will be introduced to relevant objectives that require a very specific set of skills, and the curated learning materials that will help you to achieve them.

- Technologies and Overall Integration Strategy
- Integration Solution Tools
- Security

Each learning resource has a related skill level: **Beginner**, **Intermediate**, or **Advanced**. Resources marked **Core** cover essential concepts, while those marked **Recommended** provide additional materials for further edification.

1. Technologies and Overall Integration Strategy

This section provides an overview of tools and methods for designing effective integration strategies.

1.1 Describe the Enterprise Integration landscape and overall integration strategy, detailing associated risks, trade-offs, and business and technical considerations within a customer environment.

Integration Architecture for the Salesforce Platform

This post shares patterns gathered from my experience in creating an effective Integration Architecture for clients, and shares a reference design drawn from many of these.
Tags: Beginner, Core

1.2 Awareness of data backup/archiving/data warehousing integration strategies.

Salesforce Backup and Restore Essentials Part 1

This article provides in-depth details on core backup concepts, Salesforce APIs used for backup processes, and how to benchmark backup performance and ensure your backups are running as smoothly as possible.
Tags: Intermediate, Recommended

Salesforce Backup and Restore Essentials Part 2

This article discusses some additional benefits to having a backup & restore process that aren't directly related to disaster recovery, such as using backup/restore in your release management process, and demonstrates ways to increase your overall return on investment that you might not have even considered.
Tags: Intermediate, Recommended

1.3 Recommend and justify the appropriate integration strategy and the use of common integration patterns.

Patterns and Best Practices for Enterprise Integration

This site is dedicated to making the design and implementation of integration solutions easier.
Tags: Beginner, Recommended

Integration: Mashups and Composite Applications

Learn how mashups and composite applications allow you to integrate at the user interface (UI) layer.
Tags: Beginner, Recommended

Integration: Introduction to the APIs

Find out about the variety of APIs available on the Force.com platform and common security considerations.
Tags: Beginner, Recommended

Integration: The Web Service API

Get introduced to the basics of the SOAP-based Web Services API.
Tags: Beginner, Recommended

Integration: The REST-based APIs

Learn the basics of the REST-based APIs.
Tags: Beginner, Recommended

Integration: The Bulk API

Find out about the basics of the Bulk API, which is used to handle large data volumes.
Tags: Beginner, Recommended

Integration: Outbound Messaging

Learn the basics of Outbound Messages, an asynchronous tool for sending messages from the Force.com platform.
Tags: Beginner, Recommended

Integration: Salesforce to Salesforce

Learn how to set up Salesforce to Salesforce for communication between your Salesforce organizations.
Tags: Beginner, Recommended

Integration Characteristics

Review key integration characteristics that will help you choose an appropriate Salesforce integration mechanism.
Tags: Beginner, Recommended

Salesforce Integration Mechanisms

Get an overview of the key integration mechanisms available when you are integrating with Salesforce and the Force.com platform. Examine the key features of the main APIs, built-in declarative mechanisms, portal technologies, and tools and toolkits.
Tags: Beginner, Recommended

1.4 Provide details on the integration components involved in a flow, describe Salesforce integration patterns, and discuss considerations pertinent to transaction management and error and exception handling.

Integration Patterns and Practices

This document describes strategies (in the form of patterns) for common integration scenarios. Each pattern describes the design and approach for a particular scenario, rather than a specific implementation.
Tags: Advanced, Core

RESOURCE: See “Suggested Activities,” later in this document.

2. Integration Solution Tools

This section provides an in-depth look at specific tools to integrate other applications with the Force.com platform.

2.1 Recommend and justify the appropriate platform-specific integration technology used to integrate with external systems and describe the capabilities, limitations, and trade-offs.

Salesforce Limits Quick Reference Guide

This guide provides commonly referenced limits for Salesforce. This guide may not cover all limits or may contain limits that don't apply to your organization. Stated limits aren't a promise that the specified resource is available at its limit in all circumstances.
Tags: Beginner, Core

An Introduction: Integrating with the Salesforce Force.com Platform

This article outlines a few of the options available for integrating other client applications with the Force.com platform. The main focus is on Web Service integration, including both inbound and outbound to Force.com.
Tags: Beginner, Core

CTI Toolkit

Open CTI allows partners and customers to embed third-party, web-based call control and softphone tools directly inside the salesforce.com user interface. Salesforce.com additionally provides a cross-domain JavaScript API to enable partners to perform channel actions such as screen pops, saving call logs, enabling click to dial, etc.
Tags: Intermediate, Recommended

2.2 Describe the capabilities and limitations of the Force.com integration APIs and determine the appropriate approach.

Which API Should I Use?

Salesforce provides programmatic access to your organization's information using simple, powerful, and secure application programming interfaces.
Tags: Beginner, Core

Best Practices to Avoid Excessive SOAP and REST API DML

When developing integration applications using the Force.com SOAP API or REST API, you should make sure your code is as efficient as possible to avoid poor performance or API limit issues due to excessive API calls. See the following best practices for tips on making your integration code as efficient as possible.
Tags: Beginner, Core

Salesforce provides two WSDL files, what are the differences?

Salesforce provides a WSDL (Web Service Description Language) files. They are called "Enterprise WSDL" and "Partner WSDL".
Tags: Beginner, Core

Setting Up Your Salesforce.com Web Services API Applications

A WSDL document is an XML file that describes the format of messages you send and receive from a Web service. It's the protocol that your development environment's SOAP client uses to communicate with external services like Salesforce.com.
Tags: Beginner, Recommended

2.3 Leverage the features and capabilities of the platform appropriately within the solution.

Mashups: The What and Why

This first article provides an introduction to the core concepts found in mashups, examining the background, techniques, and consequences of using, creating, and enabling mashups in your IT infrastructure.
Tags: Beginner, Core

Integrating with Outbound Messaging

Learn how Outbound Messaging, a point-and-click mechanism, can be used to send SOAP messages containing data to external systems for integrations. The course will introduce the benefits of this integration method, such as the retry mechanism, as well as best practices for its implementation. You will be guided through the steps of setting up a workflow to send an outbound message and of monitoring the state of the message. You will also learn the requirements for setting up the endpoint that will receive the messages.
Tags: Beginner, Recommended

Mashups and Visualforce

Learn how mashups can be implemented through standard page layouts as well as through Visualforce pages and components. This will include an introduction to Visualforce, Salesforce's component-based framework for building and deploying custom user interfaces. Finally, you'll experience mashups in action both as links to an external system and as embedded Visualforce pages.
Tags: Intermediate, Recommended

Force.com Canvas

Explore Force.com Canvas, a more powerful framework that allows you to integrate Web-based applications into the Salesforce user interface and that facilitates a deeper level of integration between the application and Salesforce through the Canvas JavaScript SDK. Watch a demonstration of how to use the Canvas Framework to embed an external application within a Salesforce page and how to use the JavaScript SDK to both read from and write to Salesforce from the external application.
Tags: Intermediate, Recommended

Integrating with Force.com: JSON Basics

Learn how JSON–which is used by the Force.com REST, Chatter REST, and Streaming APIs–structures data. Examine some of the key methods you will need to serialize and deserialize JSON in either Java or C#.
Tags: Intermediate, Recommended

Integrating with the Force.com Streaming API

Learn how to use the Streaming API to send notifications to external applications when key changes occur to your Salesforce data. You'll start by learning about the publish/subscribe model used by the API and how it relies on long polling using CometD and the Bayeux protocol. You'll then explore how to configure the parameters that determine what data is sent by the API and when it will be sent. You'll use Workbench to practice what you've learned and see the API in action. Finally, you'll review topics that should be considered when implementing your solution.
Tags: Intermediate, Recommended

SOAP API Architecture

Learn about SOAP architectures and the capabilities of the Force.com SOAP API. Understand how the API processes large data volumes and learn about the security mechanisms it provides, as well as the limits placed on its use.
Tags: Intermediate, Recommended

Integration Using the SOAP API

Learn how to obtain a WSDL document from your Salesforce organization that can be used to connect to and interact with that organization. Learn and practice how to connect to your organization, how to get data from your organization and process the results, and how to perform data manipulation methods using the enterprise WSDL.
Tags: Intermediate, Recommended

Working with the Partner WSDL

Learn and practice how to use the SOAP API partner WSDL to work with your organization. Understand its advantages and how it differs from the enterprise WSDL.
Tags: Intermediate, Recommended

Advanced SOAP API Topics

Learn about specialized SOAP API methods and advanced topics that facilitate the design and implementation of programs that use the SOAP API, such as debugging tools and design patterns.
Tags: Intermediate, Recommended

An Introduction to the Salesforce REST APIs

Learn the basics of the REST architecture and how it is used in both the Force.com REST API and Chatter REST API.
Tags: Intermediate, Recommended

The Force.com REST API

Learn and practice how to connect to your organization, how to query for data, and how to perform data manipulation using the REST API.
Tags: Intermediate, Recommended

The Chatter REST API

Explore how the specialized Chatter API supports building social applications quickly and effectively.
Tags: Intermediate, Recommended

Integrating with the Force.com Bulk API

Learn about how to use the Force.com Bulk API to move large data sets into and out of your Salesforce organization. You'll start by learning when to use the API and exploring the architecture on which it is based. You'll then learn about the code necessary to request that Salesforce perform a DML or query operation, how you can monitor your requests, and how the results of your requests will be returned.
Tags: Intermediate, Recommended

General Guidelines for Data Loads

Planning Bulk API data loads for optimal processing time.
Tags: Advanced, Recommended

RESOURCE: See "Suggested Activities," later in this document.

3. Security

This section covers security protocols for securing both inbound and outbound integration on the Force.com platform.

3.1 Describe the risks and impacts when designing an integration with a cloud-based system.

About Salesforce Certificates and Keys

Salesforce certificates and key pairs are used for signatures that verify a request is coming from your organization.
Tags: Beginner, Recommended

3.2 Describe how security requirements are met at each of the integration layers.

Integrating with Force.com: Security

Learn about Salesforce security mechanisms for securing both inbound and outbound integrations.
Tags: Beginner, Recommended

3.3 Secure an integration inbound to Salesforce.

Connected Apps Overview

This webpage provides an introduction to Salesforce Connected Apps and their use to provide application-specific permissions.
Tags: Intermediate, Core

Digging Deeper into OAuth 2.0 on Force.com

This article takes an in-depth look at the OAuth 2.0 protocol in the context of Force.com, and is intended for developers and architects with an understanding of security and identity concepts, such as authentication and authorization.
Tags: Intermediate, Recommended

3.4 Secure an integration outbound from Salesforce.

Configuring Remote Settings

Before any Visualforce page, Apex callout, or JavaScript code using XmlHttpRequest in an s-control or custom button can call an external site, that site must be registered in the Remote Site Settings page, or the call will fail.
Tags: Beginner, Core

Named Credentials Overview

A named credential specifies the URL of a callout endpoint and its required authentication parameters in one definition. You can simplify the setup of authenticated Apex callouts by specifying a named credential as the callout endpoint.
Tags: Intermediate, Core

RESOURCE: See “Suggested Activities,” later in this document.

Suggested Activities

To practice these activities, you may do one of the following:

- Request a free Practice Org by creating a case here.
  - Question Type: Architect Support
  - Question Detail: Request Practice Org

  You should receive login information in about two business days.
- Use your existing Developer org.
- Sign up for a free Developer Edition account here.

1. Workflow

Build a Workflow to trigger an Outbound Message on update to an Account record.

- Send the outbound message to a simple HTTP-capture service (e.g., http://requestb.in/).
- Examine the delivered messages.
- Observe the automated retry.
- Check the Outbound Message queue and cancel the failed message.

2. SOAP Integration

Using a simple SOAP integration tool (e.g., SOAPUI, Eclipse), import the SOAP WSDL and perform each of the core operations (login, query, upsert, etc.) to understand the message structure.

3. REST Integration

Using a simple REST integration tool (e.g., cURL, Postman), authenticate and perform some basic GET and POST requests to the REST API to understand the message structure.

Build Materials

1. Fire and Forget

2.3 Leverage the features and capabilities of the platform appropriately within the solution.

Use Case

A company, Universal Containers, uses several systems as part of their enterprise system landscape, including Salesforce.com and an ERP system. Salesforce.com is the master system for CRM data, such as Accounts and Opportunities. The ERP system is the master system for customer orders and shipping and billing information.

Detailed Requirements

As part of their business process flow, once a sales opportunity is successfully closed ("Closed Won") an order needs to be created with the details from the opportunity for shipping and subsequent invoicing. The opportunity is maintained in Salesforce.com, but the order needs to be created in the on-premise ERP system, as it is the order master. Once the order is created in the ERP system, the opportunity status must be changed to indicate that the order was created.

Assumptions

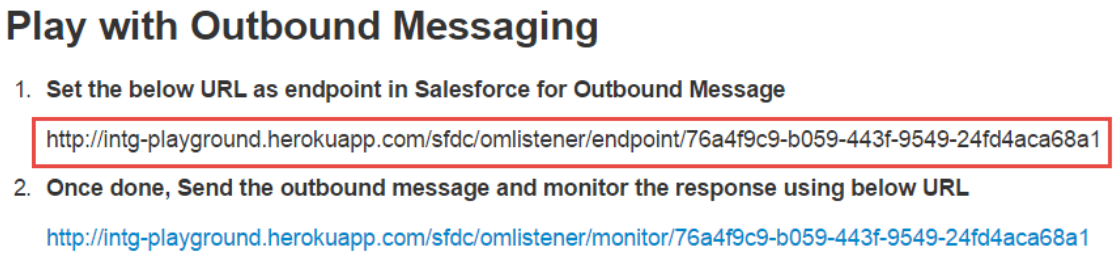
- Universal Containers has middleware that integrates with the on-premise ERP system.
- This middleware can host a SOAP endpoint using a WSDL from Salesforce.
- The user doesn't need to be immediately notified of the order number after the opportunity converts to an order.
- The solution must only use declarative features of Salesforce.com.

Prerequisite Setup Steps

Get a proxy endpoint for a proxy service (Outbound Message Listener):

1. Go to the Integration Playground for Outbound Messaging here: http://intg-playground.herokuapp.com/sfdc/omlistener

2. Copy the unique endpoint that is randomly generated. This helps to uniquely monitor the messages coming in from an outside system.



3. Open the monitor URL in a new tab. This helps to monitor all the messages coming to this endpoint and provides an ability to play around with the responses being sent.

Considerations

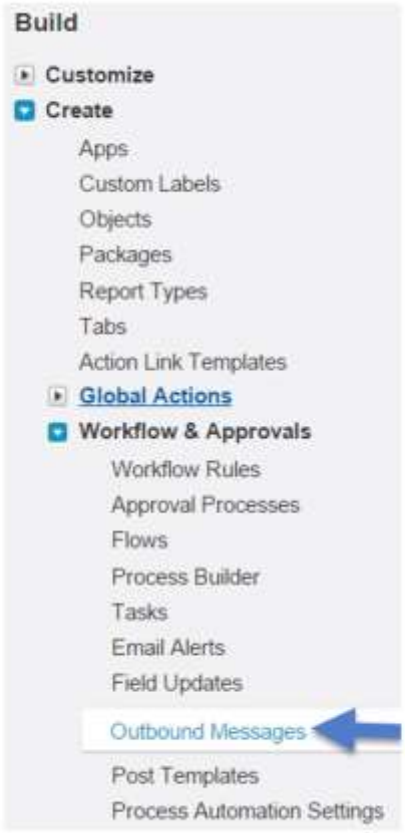
- What are the patterns that can be used?
- What is the integration frequency: real-time vs. batch?
- What are the Salesforce.com platform limits?
- How are errors handled both in Salesforce.com and on the third-party platform?
- Can you send information from multiple related objects?
- Which web-services protocols can be used?
- How is security handled (2-way SSL, authentication)?

Solution

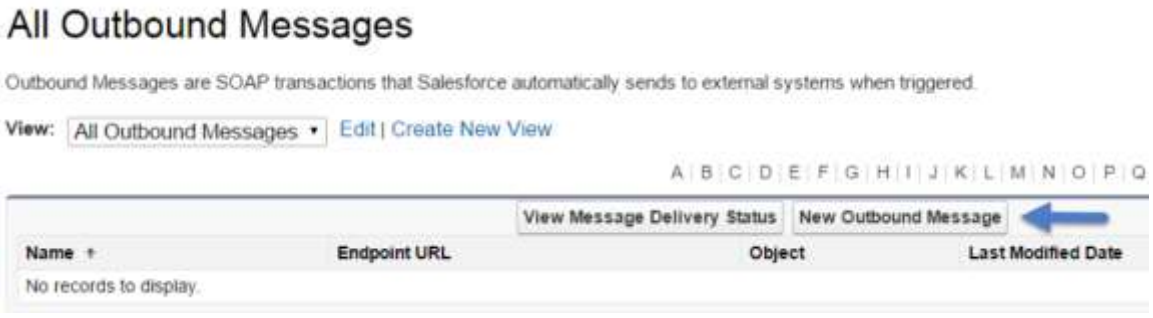
Best Solution Overview

Log in to the Salesforce org with your credentials.

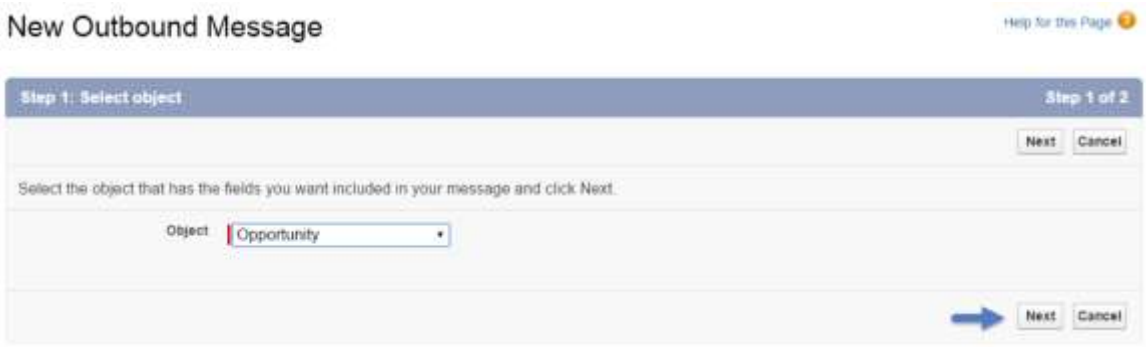
1. **Create a new outbound message.**
   - From Setup, click **Create | Workflow & Approvals | Outbound Messages**.



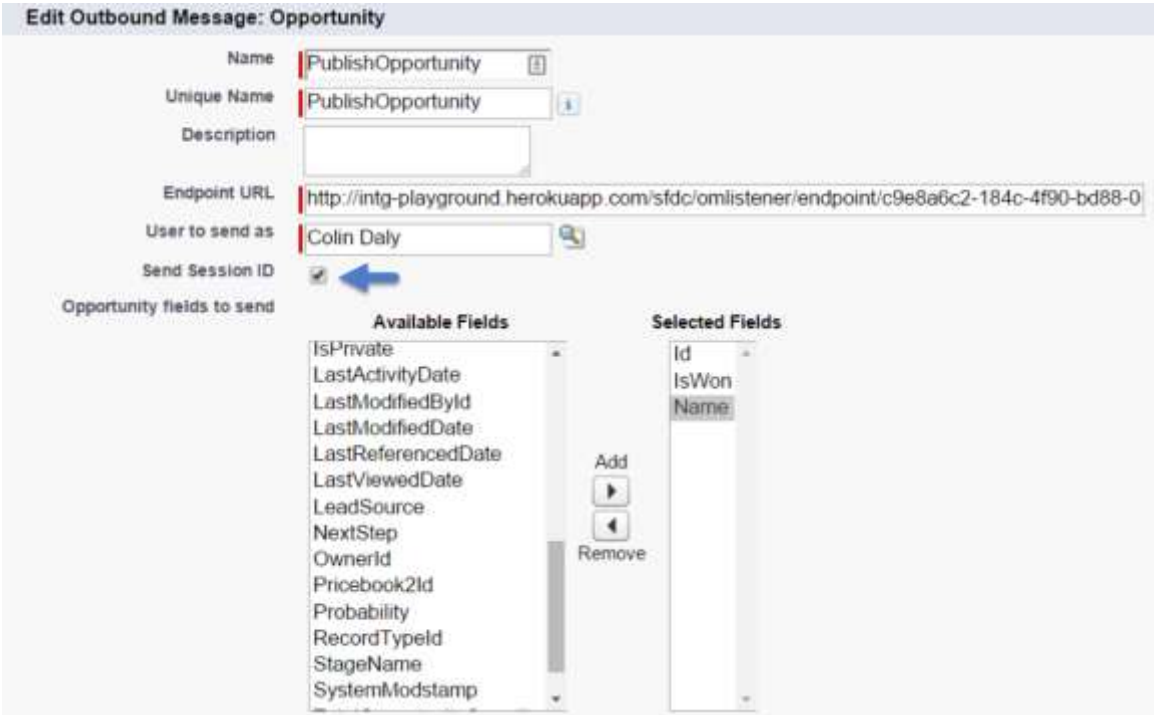
   - Click **New Outbound Message**.



- Choose the object that has the information you want included in the outbound message, in this case **Opportunity**, and click **Next**.

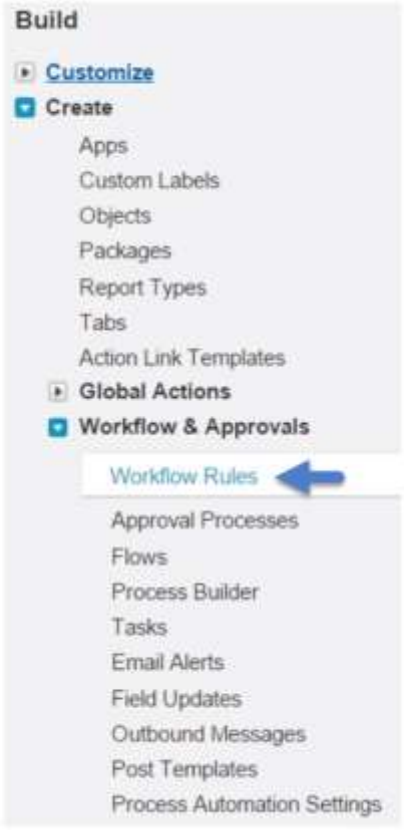


- Enter a name and description for this outbound message.
- Paste the endpoint URL that was copied in prerequisite step.
- Check **Send Session ID**.



- Choose fields to be sent in the outbound message and click **Save**.

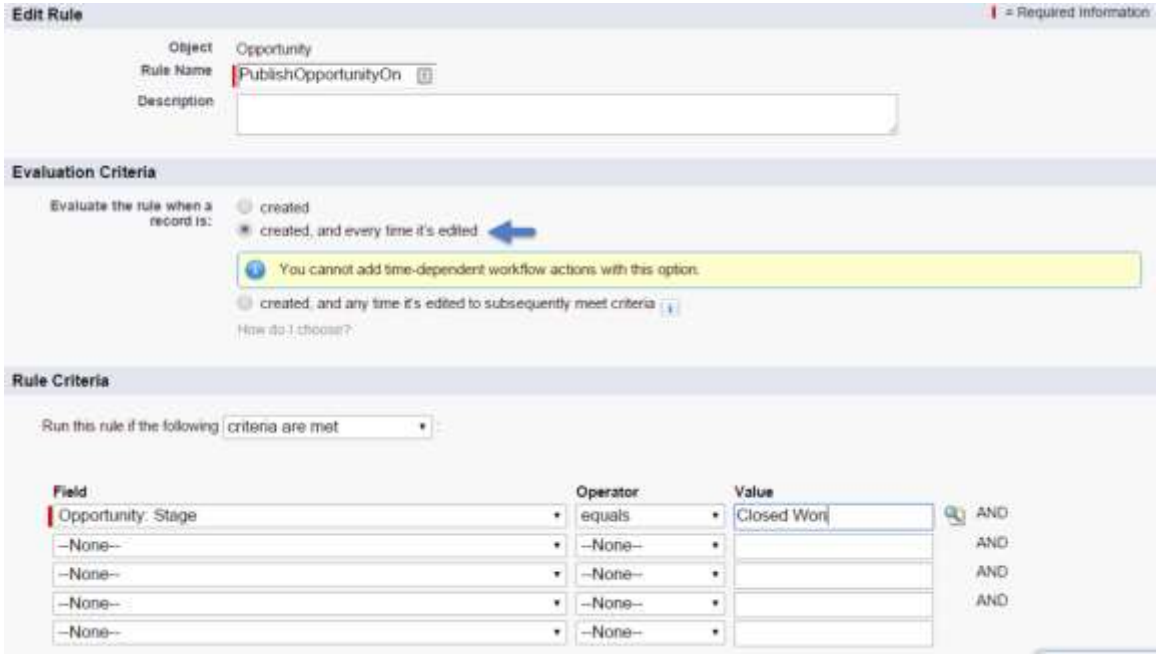
2. **Set up a Workflow rule.**
   - From Setup, click **Create | Workflow & Approvals | Workflow Rules**.



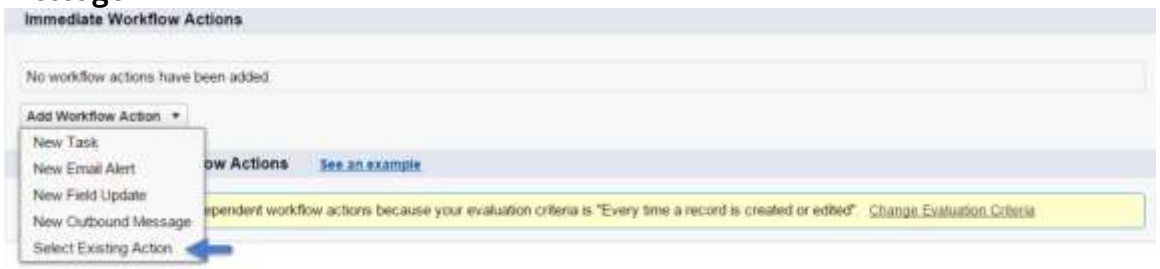
   - Click **New Rule**.

   - Choose the object that has the information you want included in the outbound message, in this case **Opportunity**, and click **Next**.

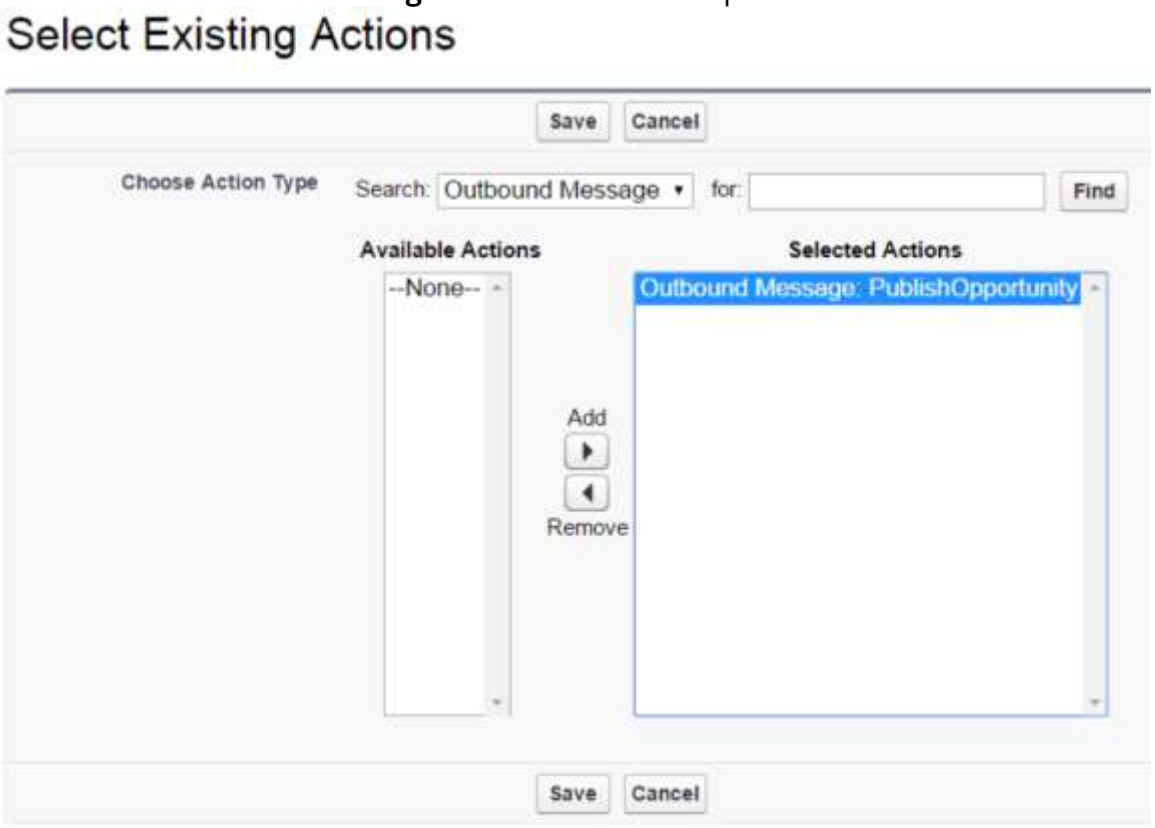
- Enter the Rule Name, choose the Evaluation Criteria **created, and whenever it's edited**, and set the Rule Criteria: in this case, **Opportunity: Stage equals Closed-Won**.



- Click **Save & Next**.
- Click **Add Workflow Action | Select Existing Action | Select Outbound Message**.



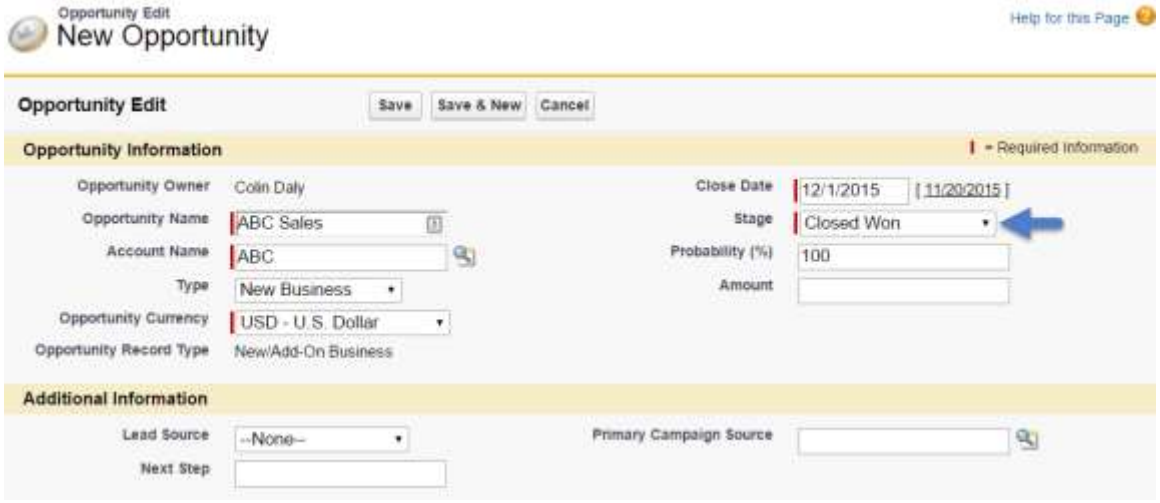
- Choose the **Outbound Message** action created in step 2. Click **Save**.



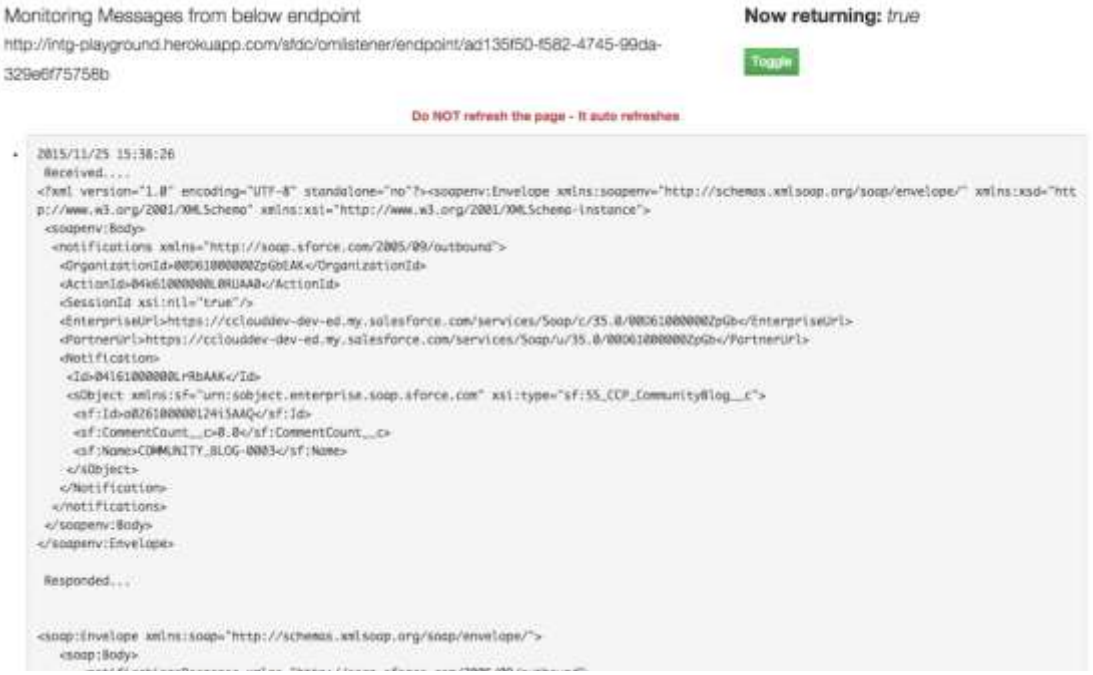
- Click **Done** and then click **Activate**.

3. **Play with outbound messaging.**
   - Now that the setup is complete, it's time to test and play with outbound messaging.
   - Go to the monitor URL from step 1. Make sure the page reads "Now returning: true." That means the endpoint acknowledges outbound messages with success. By clicking the **Toggle** button, it can be switched to return false, which means that the endpoint acknowledges outbound messages with failure.

- Go to Salesforce and create a new opportunity with the stage **Closed Won**.

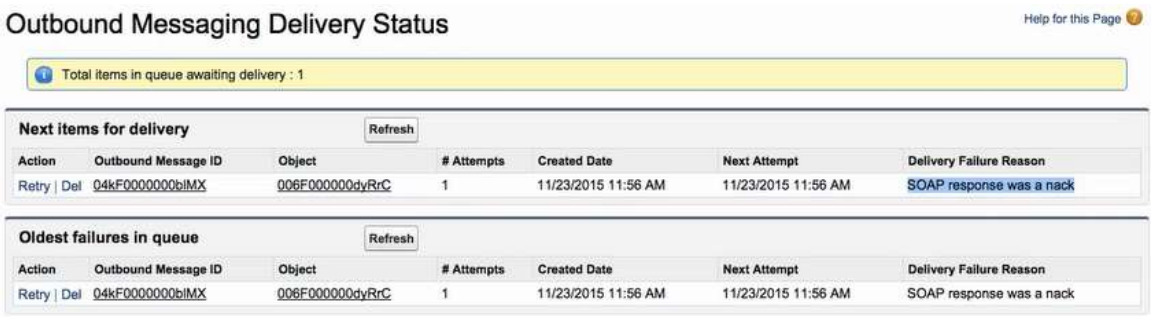


- Switch to the monitor URL and you should see messages being received by endpoint.



Alert: Please have only one browser listener window open at any point of time.

4. **Play with error scenarios.**
    - Go to the Monitor URL and click the **Toggle** button to make the endpoint return false. This will throw an error back to the Salesforce outbound message, which can be monitored and retried from Salesforce.
    - After toggling the endpoint to return false, go to Salesforce and create a new opportunity with the stage as **Closed Won**.
    - Go to **Setup | Monitor | Outbound Messages**. It should display errors as shown in the screenshot below. You can also see it keeps retrying.



- Go back to the Monitor URL and toggle the response back to true.
- Now through automated retry, the messages are delivered again. It can also be manually retried by clicking the **Retry** button.

2. Request and Reply

1.4 Provide details on the integration components involved in a flow, describe Salesforce integration patterns, and discuss considerations pertinent to transaction management and error and exception handling.

Use Case

Universal Containers uses Salesforce, but has a separate system that contains customer order, billing, and shipping information. Salesforce.com is the master data source for Accounts and Opportunities.

Universal Containers wants to display the order status and billing history for a customer account without having to store that data in Salesforce because of NPI data security concerns. They have an existing REST/Web Service that can return order status and shipping information given an account number, but cannot otherwise display this data in a browser.

Prerequisite Setup Steps

Set up remote site setting: http://intg-playground.herokuapp.com/.

Considerations

- Does the call to the remote system require Salesforce to wait for a response before continuing processing? In other words, is the call to the remote system a synchronous request-reply or an asynchronous request?
- If the call to the remote system is synchronous, does the response need to be processed by Salesforce as part of the same transaction as the initial call?
- Is the message size relatively small or large?
- Is the integration based on the occurrence of a specific event, such as a button click in the Salesforce user interface, or DML-based events?

Solution

Solution Overview

In the Request-Reply pattern, Salesforce makes a call to the remote system to either perform an operation or fetch data. It then waits for successful completion of that call. To signify successful completion, the remote system synchronously replies with the requested data or a success indicator code. The remote system exposes a web service (REST or SOAP) that allows other systems, such as Salesforce, to integrate with it.

The high-level solution steps are as follows:

- Salesforce consumes the Order status and billing/shipping service from Universal Containers.
- Use Order external Id in Salesforce to fetch order status and billing information (NPI security data) from the external system.
- Create a Visualforce page and extension controller to call REST/ Web service with required parameters.
- Create the custom controller that parses the returned values from the REST callout or Web Service.
- The Visualforce page subsequently renders the order and billing details to the user.

Best Solution Overview

The steps necessary to create the integration pattern are explained in detail in the following sections.

1. **Create an Object.**

   Create an Object (iOrder) with following field and API names:

Field Name	API Name	Field Type
Order Name	Name	Text(80)
Invoice Amount	Invoice_Amount__c	Currency(18, 0)
Invoice Number	Invoice_Number__c	Text(20)
Opportunity	Opportunity__c	Master-Detail(Opportunity)
Order External ID	Order_External_ID__c	Text(20) (External ID)
Order Status	Order_Status__c	Text(20)
Product Name	Product_Name__c	Text(30)
Shipping Number	Shipping_Number__c	Text(30)

Note the use of field "External ID" to store reference to the OrderID in the external system. Furthermore, you can create your own object and appropriate relationships. If you have your own object model, use and carry the appropriate API names in Apex and Visualforce pages.

2. **Create Apex classes to invoke REST/ Web Service.**

   Create an Apex class – OrderShipInfo with getOrderShipInfo method to invoke the external REST/ Web Service.

   The getOrderShipInfo method invokes the REST/Web Service by appending Order ID to the End Point URL. It then de-serializes the JSON string received in the HTTP response to populate the ShipInfo object, which is then used by the VisualForce page to display the order status.

```
1   public with sharing class OrderShipInfo {
2      private final String EXTERNAL_END_POINT_URL = 'http://intg-
    playground.herokuapp.com/services/rest/iOrder';
3       /*
4        *  Ship Info Class to hold REST API response data
5       */
6      public class ShipInfo
7      {
8        public String id{get;set;}
9        public String status{get;set;}
10       public String comments{get;set;}
11       public String orderOwner{get;set;}
12     }
13    public ShipInfo getOrderShipInfo(String externalID){
14     try{
15         HttpRequest req = new HttpRequest();
16         req.setEndpoint(EXTERNAL_END_POINT_URL + '/' + externalID);
17         req.setMethod('GET');
18         req.setHeader('Content-Type', 'application/x-www-form-urlencoded');
19         req.setTimeout(60000);
20
21         Http http = new Http();
22         HttpResponse res = http.send(req);
23         ShipInfo theStatus = (ShipInfo)JSON.deserialize(res.getbody(),ShipInfo.class);
24         return theStatus;
25     }catch(CallOutException ce){
26         throw ce;
27     }
28     return null;
29    }
30  }
```

3. **Create an Apex Controller for use in the Visualforce page.**

```
public class iOrderControllerExtension
{
private iOrder__c myOrder;
OrderShipInfo myOrderShipInfo;
OrderShipInfo.ShipInfo theShipInfo;
private ApexPages.StandardController controller {get; set;}

public iOrderControllerExtension(ApexPages.StandardController controller) {
//initialize the stanrdard controller
this.controller = controller;
this.myOrder = (iOrder__c)controller.getRecord();
}
public void refreshOrderShipInfo()
{
myOrderShipInfo = new OrderShipInfo();
theShipInfo = myOrderShipInfo.getOrderShipInfo(this.myOrder.Order_External_ID__c);

myOrder.Order_Status__c = theShipInfo.status;
update myOrder;
}
public String getOrderStatus()
{
return theShipInfo.status;
}
public String getOrderComments()
{
return theShipInfo.comments;
}
public String getOrderOwner()
{
return theShipInfo.orderOwner;
}
}
```

4. **Create a custom Visualforce page.**

   Create a Visualforce page; for example “OrderPage” with sample code:

```
<apex:page standardController="iOrder__c" extensions="iOrderControllerExtension">
<apex:pageBlock >

<apex:pageBlockSection >
<apex:pageBlockSectionItem >
<apex:outputLabel value="Order Status" for="order_status"/>
<apex:outputText value="{!iOrder__c.Order_Status__c}" id="order_status"/>
</apex:pageBlockSectionItem>

<apex:pageBlockSectionItem >
<apex:outputLabel value="External System Order ID" for="external_id"/>
<apex:outputText value="{!iOrder__c.Order_External_ID__c}" id="external_id"/>
</apex:pageBlockSectionItem>

<apex:pageBlockSectionItem >
<apex:outputLabel value="Shipping Number" for="shipping_number"/>
<apex:outputText value="{!iOrder__c.Shipping_Number__c}" id="shipping_number"/>
</apex:pageBlockSectionItem>
</apex:pageBlockSection>

<apex:pageBlockButtons location="bottom">
<apex:form >
```

```
19  <apex:commandButton action="{!refreshOrderShipInfo}"  value="Refresh Order Status"
    id="theButton"/>
20  </apex:form>

21  </apex:pageBlockButtons>
22  </apex:pageBlock>


23  <apex:outputText value="Note: Clicking the Refresh Order Status button makes a REST API
    callout to fetch shipping information from an external application" />
24  </apex:page>
```

5. **Test the Visualforce page.**

   Place the Visualforce page on the page layout of the iOrder__c object. Upon clicking **RefreshOrderPage**, the JSON response from the REST service should be displayed. Make sure that you are able to invoke the REST/ Web Service and get the response back. Click the URL to ensure that the REST service is active and JSON response is sent:

   http://intg-playground.herokuapp.com/services/rest/iOrder/12345.

   This pattern is described in detail in Chapter 2 in the Salesforce Integration Patterns and Practices guide.

3. Remote Call-In

2.3 Able to leverage the features and capabilities of the platform appropriately within the

Use Case

A company, Universal Containers, uses several systems as part of their enterprise system landscape including Salesforce.com and an ERP system. Salesforce.com is the master system for CRM data such as Accounts and Opportunities. The ERP system is the master system for customer orders, shipping and billing information.

Detailed Requirements

As part of their business process flow, when an order in created in the ERP system, it needs to be also created in Salesforce.com in real-time.

Prerequisite Setup Steps

1. Create an integration user in Salesforce. It is best practice to create a dedicated integration user for integration needs, with the profile permission of "API only" and a non-expiring password.
2. If you are not using a dedicated integration user, then the user's profile should have the "API Enabled" permission at a minimum.

Considerations

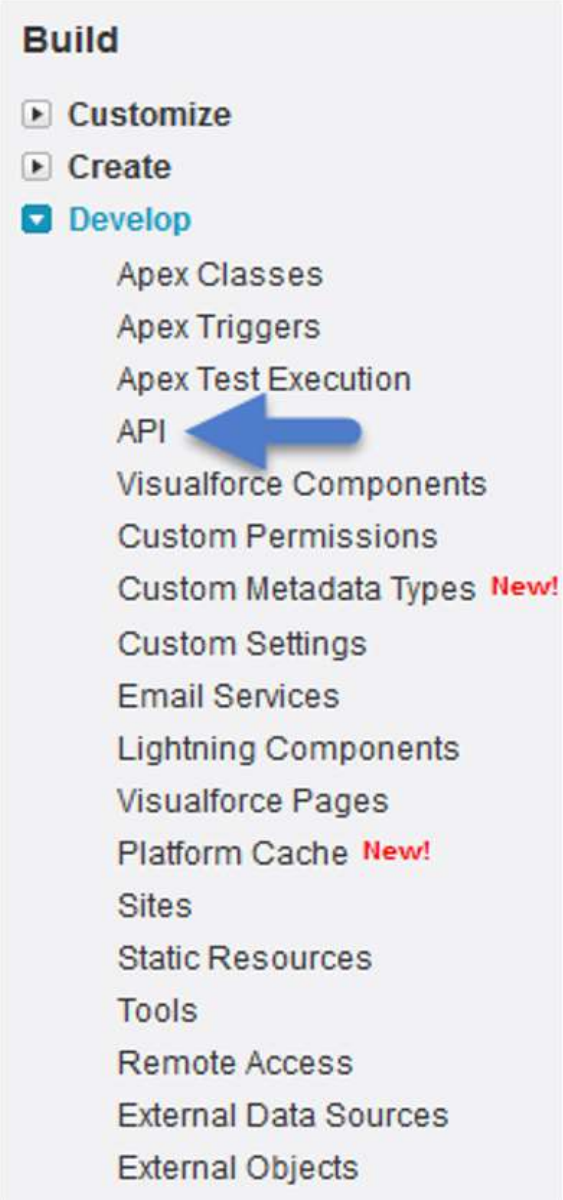
What are the patterns that can be used?
Does this need to be real-time or batch?
What are the platform limits?

Solution

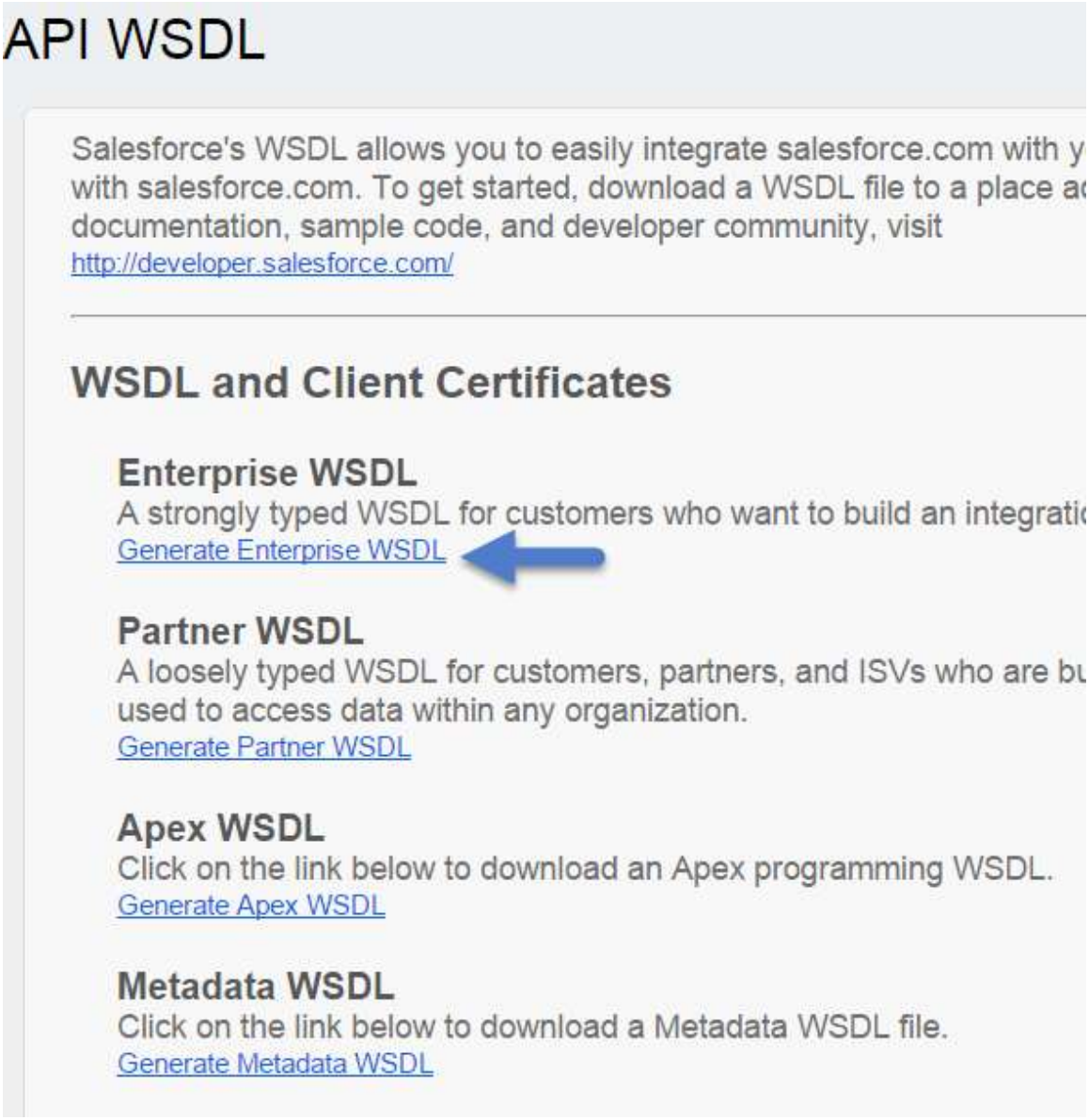
Best Solution Overview

1. Generate WSDL.

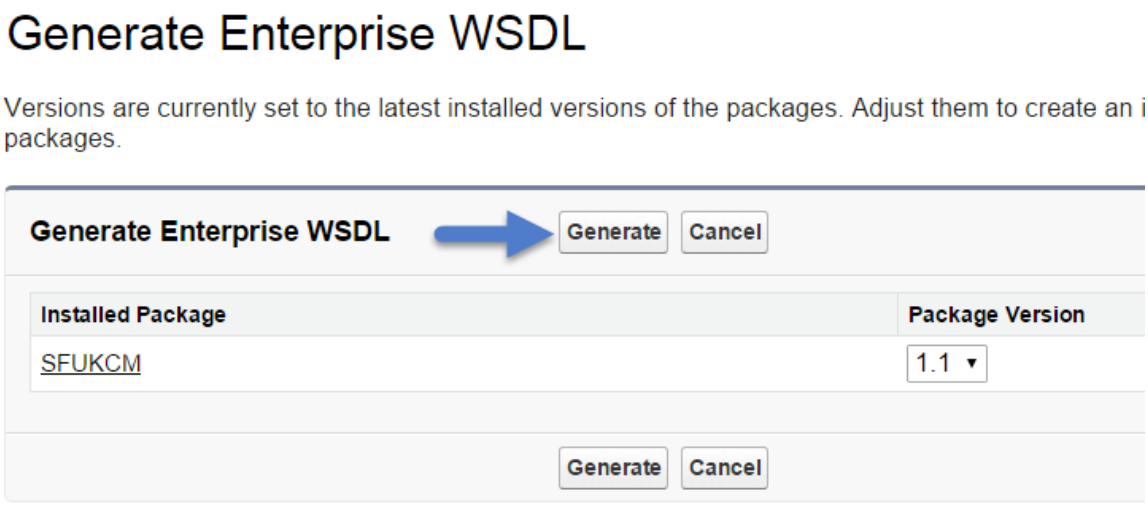
- Log in to the Salesforce Org with your credentials.
- User should have the "Modify All Data" permission to generate the WSDL.
- From Setup, click **Develop** and then **API** to display the WSDL download page.



- Download the appropriate WSDL; in this case, it's Enterprise WSDL. Click the **Generate Enterprise WSDL** link.



- If you are downloading an enterprise WSDL and you have managed packages installed in your org, click **Generate**.



- You will then be directed to a webpage displaying the document tree of an XML file. Right-click on the webpage and choose **Save as...** if you are using Chrome or **Save Page As...** if you are using Firefox. Save it to a local directory.

2. Import the WSDL into Heroku (or SOAP UI).

- Go to http://intg-playground.herokuapp.com/sfdc/inbound.
- Click the link under 1. Play with Remote call in - Enterprise WSDL.
- Click **Upload Enterprise WSDL**.
- Choose the WSDL file you downloaded.
- Upon successful upload, you should see an endpoint in the right-hand side of the page.

> Enpoint:
> https://login.salesforce.com/services/Soap/c/35.0/0DFU0000000fxWX

3. Log in to Salesforce.

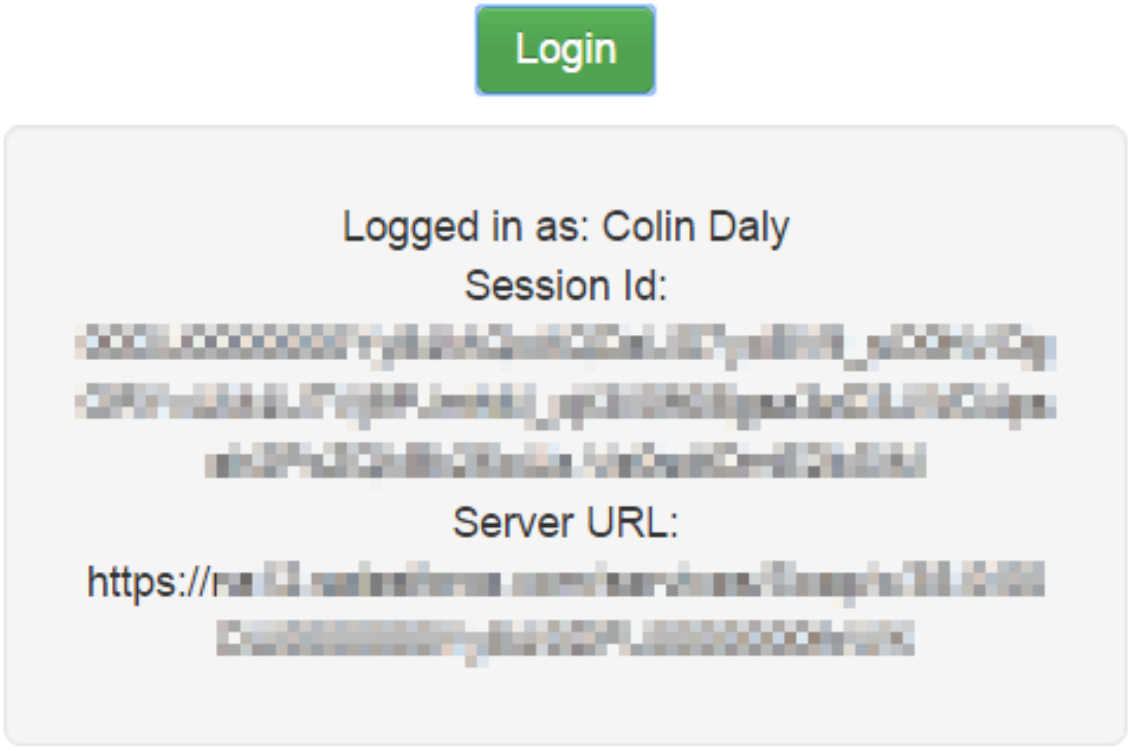
- Before doing any operation through SOAP API, a login step is necessary. This step gets the Session ID and Server URL from Salesforce, which must be passed back to Salesforce on further operations in SOAP API.
- Replace the variables in the XML data in the text area under that webpage's STEP #2 with your username and password.

STEP #2: Login to Salesforce

```
<soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
xmlns:urn="urn:enterprise.soap.sforce.com">
  <soapenv:Body>
    <urn:login>
      <urn:username>REPLACE_WITH_USERNAME</urn:username>
      <urn:password>REPLACE_WITH_PASSWORD</urn:password>
    </urn:login>
  </soapenv:Body>
</soapenv:Envelope>
```

- Click the **Login** button.

- Upon successful login, it should display your full name as defined in Salesforce, the Session ID, and Server URL.



4. **Execute SOAP Operation or, in this case, create iOrder.**
   - Copy the Session ID created in the previous step.

- Paste the Session ID into the XML data window so that it replaces the "REPLACE_WITH_SESSION_ID OBTAINED IN STEP 2" text.

STEP #2: Play with different operations in Enterprise WSDL

(Example below is a sample request of create operation for custom Object iOrder__c)

```
<!-- This message creates one record in iOrder custom object -->
<soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
xmlns:urn="urn:enterprise.soap.sforce.com"
xmlns:urn1="urn:sobject.enterprise.soap.sforce.com"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
  <soapenv:Header>
    <urn:SessionHeader>
      <urn:sessionId>REPLACE_WITH_SESSION_ID OBTAINED IN STEP
2</urn:sessionId>
    </urn:SessionHeader>
  </soapenv:Header>
  <soapenv:Body>
    <urn:create>
      <!--Zero or more repetitions:-->
```

- Click the **Submit** button.
- Upon successful post, the page will display a Success message.
- By clicking **Show Response**, the app shows the raw SOAP response returned by Salesforce.

For more details on the pattern, read Chapter 5 of Integration Patterns and Practices.

Request a Practice Org

To request a Practice Org that contains information from some of the Build Materials, please click here to open a case.

Select Question Type: Architect Support

Question Detail: Request Practice Org

ALERT: If you are not active within your practice org for 6 months, it may be deactivated.

Join the Salesforce Architect Success Group

- Want to make sure you don't miss any content release updates or news regarding the Salesforce Architect Journey?
- Looking to connect with others that have the same interest?

Click here and request to join the Salesforce Architect Success Group.