A Design Pattern and Step-by-Step Implementation Guide for Salesforce Cloud Integration through IBM Integration Bus (v9)

By

Dipak Kumar Pal (dipakpal.opentech@gmail.com)

MyOpenTech

(Salesforce Cloud Integration)

Introduction

Many enterprises have committed some extent of investment to Salesforce cloud. Majority of these Salesforce customers need to synchronize sales data - such as leads, accounts, opportunities and forecasts - with a variety of on-premises applications including ERP, CRM and other SaaS/Cloud based applications. IBM WebSphere Cast Iron Cloud integration provides the most efficient and rapid solution to Salesforce cloud integration. But organizations with fewer integration requirements, already having IBM Integration Bus (or Websphere Message Broker) in their software stack can develop Salesforce cloud integration solution without buying another expensive integration tool like Websphere Cast Iron. This article introduces a high level design pattern with step-by-step implementation guide including analysis, design and development of a generic messageflow for Salesforce cloud integration through IBM Integration Bus.

This article uses Salesforce Enterprise API WSDL for the interface design/development. However a Salesforce Partner API WSDL can be implemented with similar design/development approach. Salesforce "**Lead**" object is used for most of the examples illustrated by this article. However you can invoke all supported operations on any other object types (e.g. Account, Case, Contact) in a similar fashion.

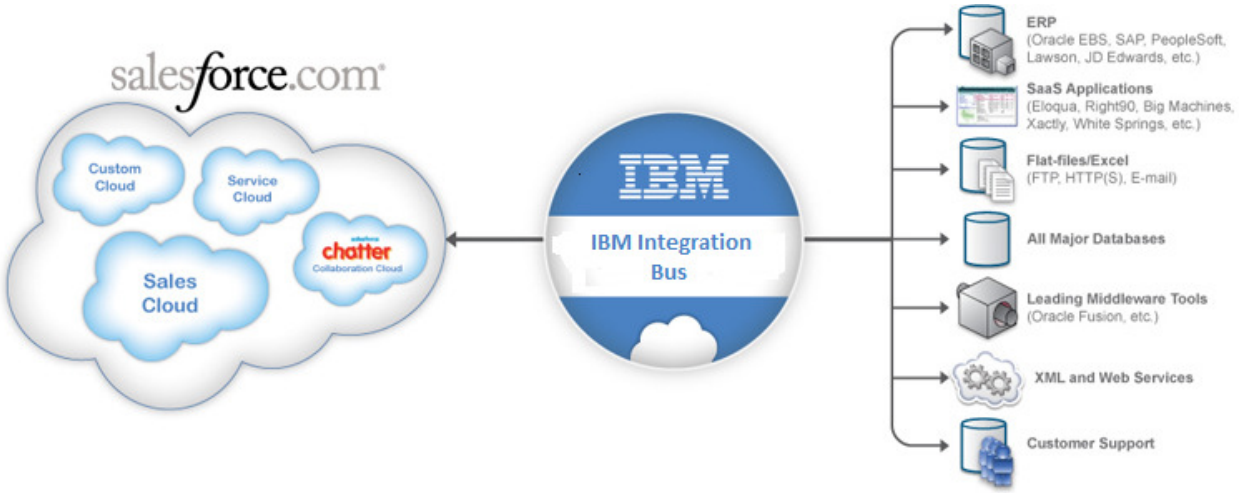

Figure-1: High Level Overview of Salesforce Cloud Integration through IBM Integration Bus

A High Level Design Pattern

Figure-2 describes a high level architectural overview of Salesforce cloud integration design pattern in IBM Integration Bus.

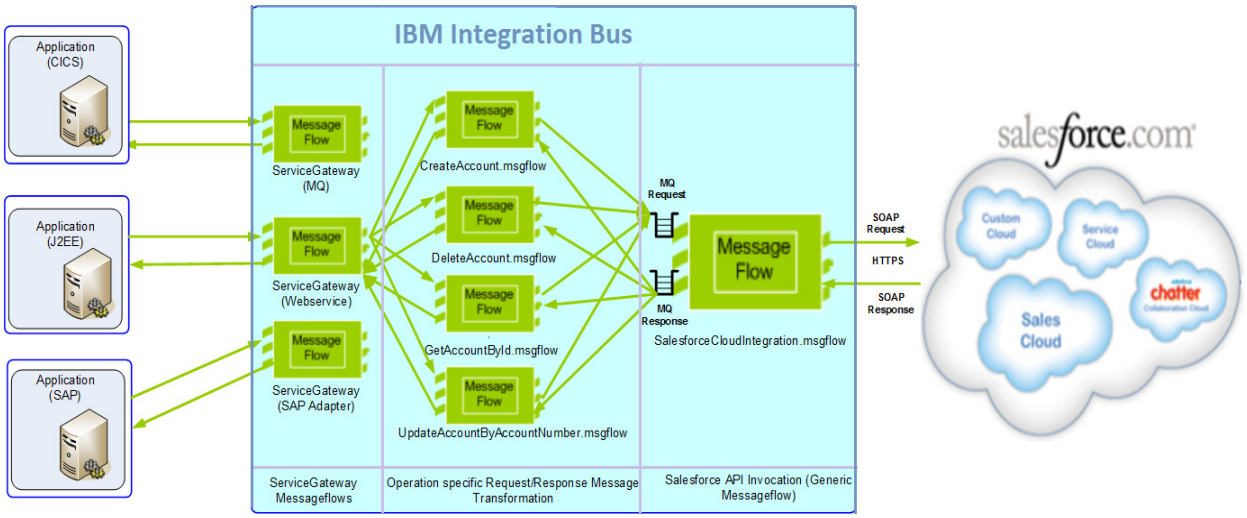

Figure-2: A High Level Architectural Overview of Salesforce Cloud Integration Design Pattern in IBM Integration Bus

This design pattern consists of following three set (or layer) of messageflows.

1. **Service Gateway Messageflows:** This set of messageflows exposes transport specific service endpoints (interfaces) to on-premises (or other cloud) applications which need to be integrated with Salesforce cloud, and routes incoming messages to operation specific messageflows responsible for data transformation. Upon getting a response back from operation specific messageflow, corresponding Service Gateway messageflow sends the response message back to the original service consumer. Integration Architect can leverage an asynchronous model of request-reply pattern to boost the throughput.

2. **Operation specific Request/Response Message Transformation Messageflows:** This set of messageflows transforms incoming messages from various on-premises (or cloud) applications into desired XML format, and invokes the generic messageflow which handles Salesforce API calls. Upon getting a response back from generic messageflow (**SalesforceCloudIntegration.msgflow**), corresponding messageflow in this layer transforms this response XML into end-application specific response message and reply back to appropriate Service Gateway messageflow for final delivery to the original service consumer. Integration Architect needs to identify list of operations (e.g. createAccount, getAccountById, getContactDetails etcs.) satisfying the organization's requirement, and design/develop a messageflow of this category for each operation exposed to service consumers (i.e. on-premises applications).

3. **A Generic Messageflow for Salesforce API invocation:** This is a common messageflow which invokes Salesforce APIs.

Detail discussion of first two set of messageflows is beyond the scope of this article. Assuming anticipated readers of this article are well familiar with integration solution design/development in IBM Integration Bus, this article primarily focuses on development of the generic messageflow (**Name: SalesforceCloudIntegration.msgflow**) for Salesforce API invocation. Rest of this article demonstrates a step-by-step implementation guide to analyze, design, development and testing of this generic messageflow including required setup at Salesforce side.

Step-1: Creation of Salesforce User (Developer Edition) Account

Create a user account at Salesforce developer site (https://developer.salesforce.com/). **Figure-3** shows a screen-shot of Salesforce user (Developer Edition) signup process.

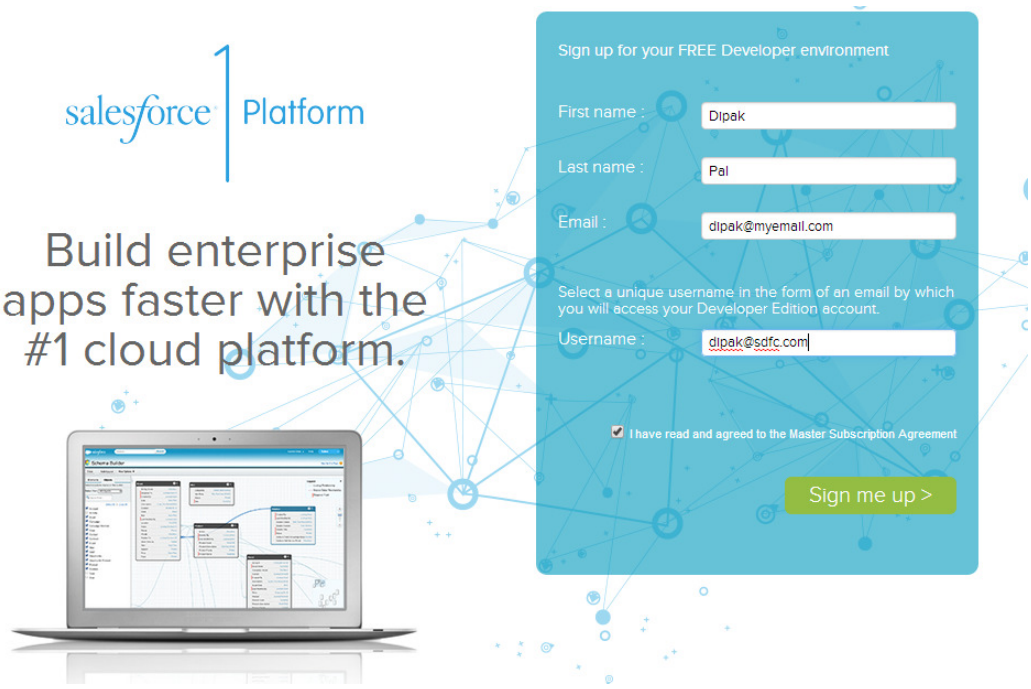

Figure-3: Screen-shot of user account (Developer Edition) creation in Salesforce developer site.

Step-2: Reset Salesforce Security Token

Login to Salesforce login page (https://login.salesforce.com/) by entering the credentials obtained during signup process, and reset the Security Token. **Figure-4** represents a screen-shot of Salesforce login page and **Figure-5** describes how to reset the Salesforce Security Token. Once you reset the Security Token, it will be emailed to your user account's email address provided during signup process.

Figure-4: Salesforce login page

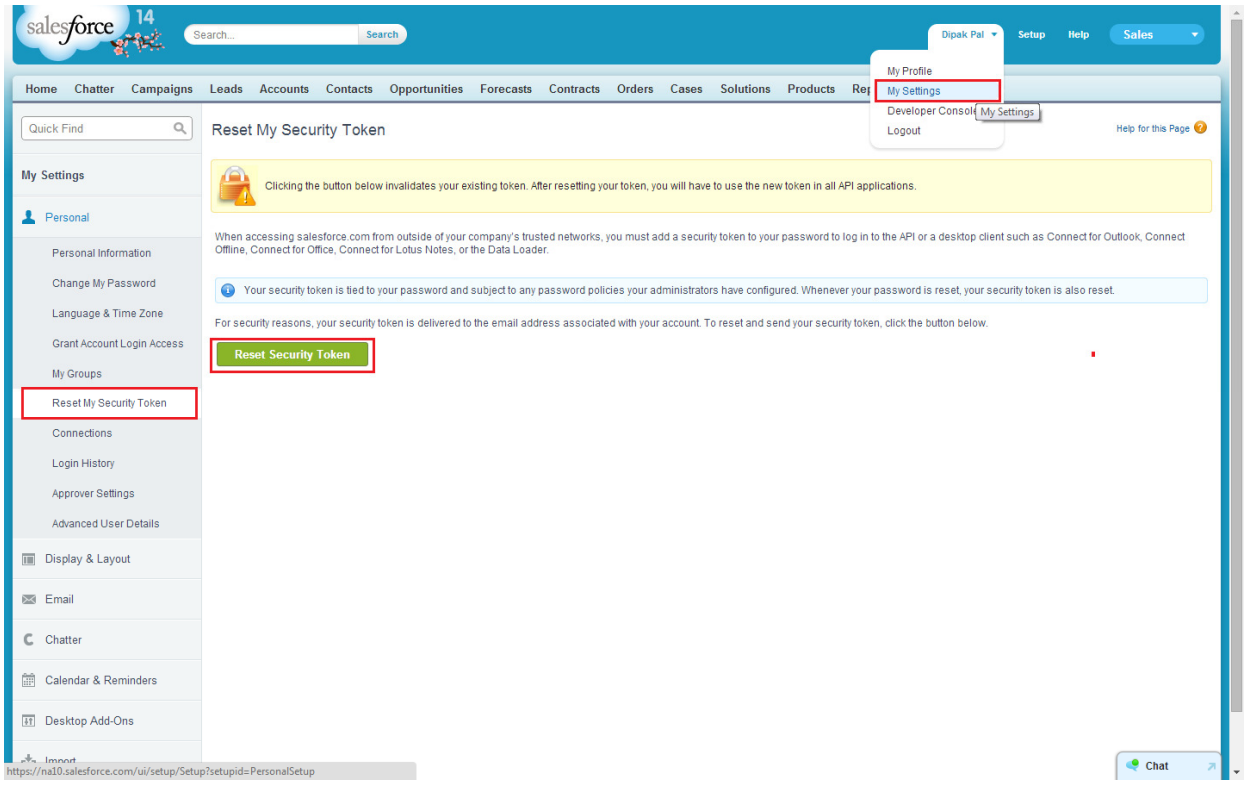

Figure-5: Screen-shot describing how to reset Salesforce Security Token

While invoking the "**login**" service exposed by Salesforce API (WSDL), actual login password and this Security Token are concatenated together to form the final password.

As an example:
Actual login password: yourpassword
Security Token: PfMSYqFyuGkzezvkG6tGdNus
Final password: yourpasswordPfMSYqFyuGkzezvkG6tGdNus

Step-3: Generate Salesforce API WSDL

Figure-6 describes how to generate Salesforce Enterprise API WSDL. If you are using Partner API for your organization, generate Partner API WSDL.

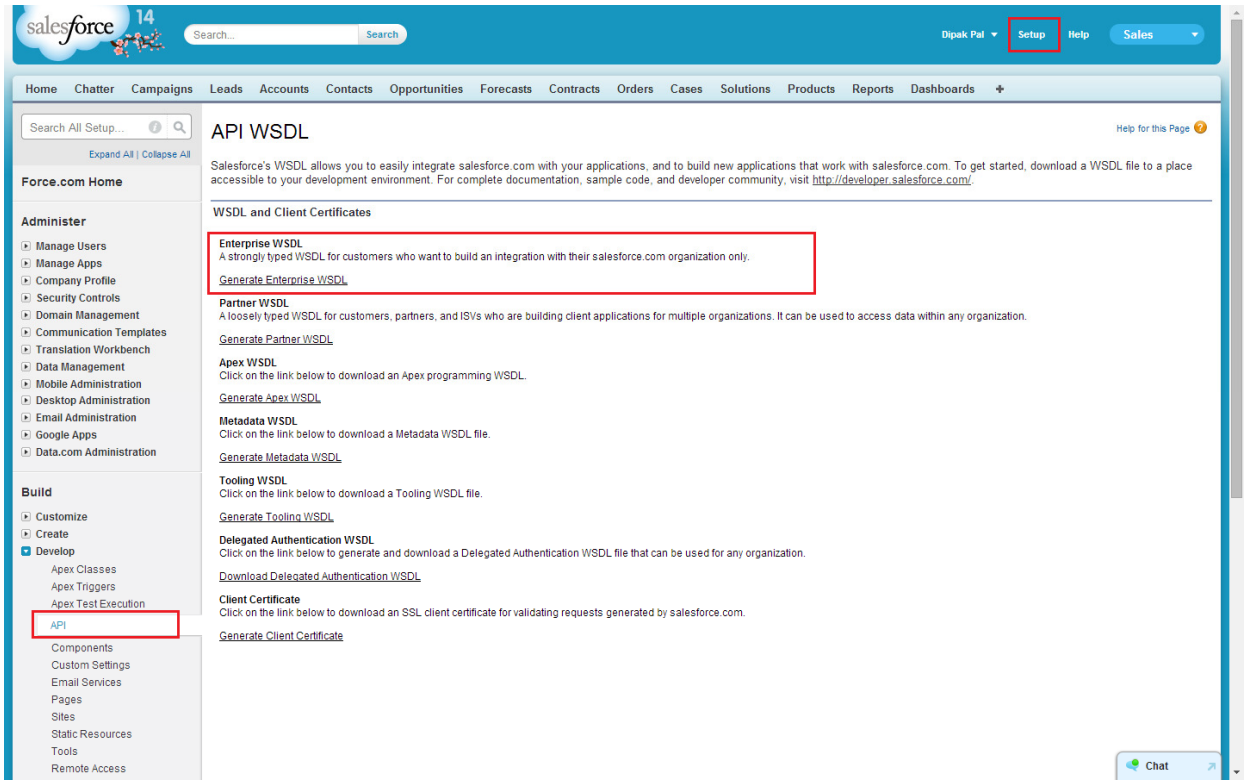

Figure-6: Screen-shot of Salesforce API WSDL generation

Step-4: Analyze Salesforce Object Types and prepare Message Schemas/Transformation Data Sheets.

This article uses SOAPUI tool for analyzing Salesforce Object Types (e.g. Account, Case, Contact, Lead). Download and install SOAPUI tool (http://sourceforge.net/projects/soapui/files/). Create a new SOAP project and import the Salesforce API WSDL generated at previous step (**Step-3**). Invoke "**login**"

operation to obtain Salesforce login SessionId and ServerUrl. **Figure-7** shows a SOAPUI tool screen-shot having "**login**" operation call.
Salesforce end-point URL for "**login**" operation: https://login.salesforce.com/services/Soap/c/30.0 .
However you can use Salesforce Sandbox endpoint URL (https://test.salesforce.com/services/Soap/u/29.0) if your organization is a Salesforce customer and you are using Salesforce Partner API WSDL.

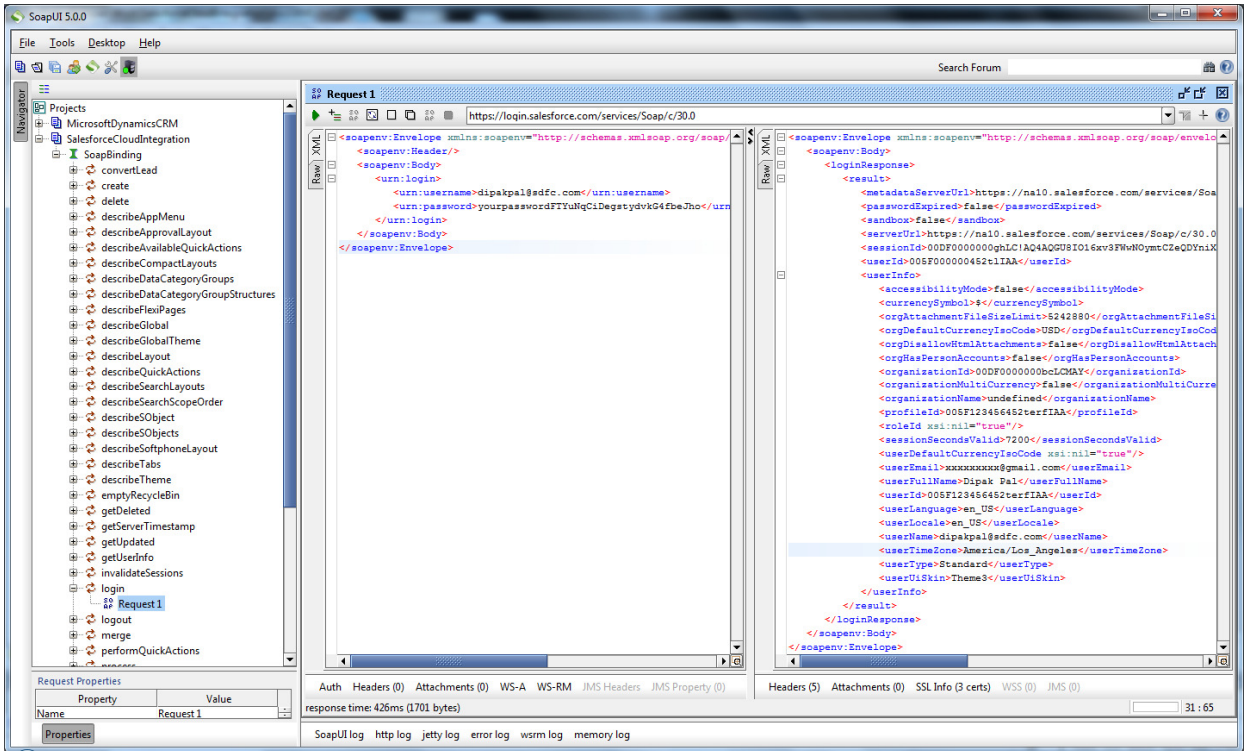

Figure-7: Salesforce "**login**" operation call at SOAPUI tool

```
<soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/" xmlns:urn="urn:enterprise.soap.sforce.com">
  <soapenv:Header/>
  <soapenv:Body>
    <urn:login>
      <urn:username>yourusername</urn:username>
      <urn:password>yourpasswordFTYuNqCiDegstydvkG4fbeJho</urn:password>
    </urn:login>
  </soapenv:Body>
</soapenv:Envelope>
```

Table-1: Sample SOAP request XML for Salesforce "**login**" operation.

Invoke "**describeSObject**" operation to get API Field details of desired Salesforce Objects (e.g. Lead for this article). Use <**serverUrl**> from "**login**" response message as Webservice endpoint URL for "**describeSObject**" operation. And use <**sessionId**> from "**login**" response message as "***soapenv:Envelope/soapenv:Header/urn:SessionHeader/urn:sessionId***" field value at

"**describeSObject**" request SOAP message. **Figure-8** shows SOAPUI tool screen-shot having "**describeSObject**" operation call and **Table-2** contains a sample SOAP request XML.

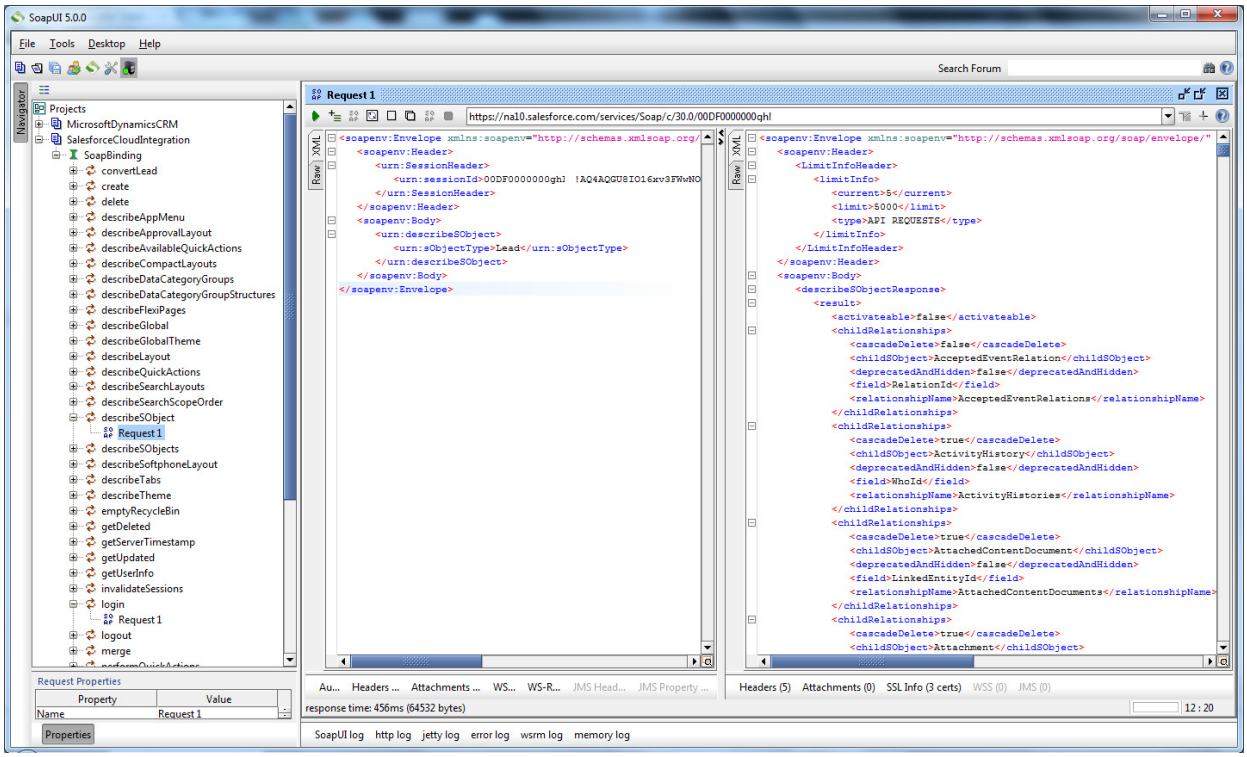

Figure-8: SOAPUI screen-shot of "**describeSObject**" operation call

```
<soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/" xmlns:urn="urn:enterprise.soap.sforce.com">
  <soapenv:Header>
    <urn:SessionHeader>
<urn:sessionId>00DF0000000ghBC!AQ4AQGU8IO16xv3FWwNOymtCZeQDYniX5EYBO1VZhrJbvzpniwL.ttM46uUzLanAOyTQjohq6zjdiC
WtgljLTn5qxcAuWwcX</urn:sessionId>
    </urn:SessionHeader>
  </soapenv:Header>
  <soapenv:Body>
    <urn:describeSObject>
      <urn:sObjectType>Lead</urn:sObjectType>
    </urn:describeSObject>
  </soapenv:Body>
</soapenv:Envelope>
```

Table-2: Sample SOAP request message for "**describeSObject**" operation

Use any XPATH evaluation tool to analyze "**describeSObject**" response XML and get list of API field names, data types and other information if required. This article uses **XPathBuilder** tool (http://www.bubasoft.net/product/xpath-builder/) to evaluate XPATH expressions. **Figure-9** represents a screen-shot of **XPathBuilder** tool displaying list of API fields for Salesforce "**Lead**" object. Some API fields have constraints with enumerated list of values (**picklist**) which also can be obtained from

"**describeSObject**" response XML. **Figure-10** shows a screen-shot of **XPathBuilder** tool displaying list of picklist values for "**Industry**" field at "**Lead**" API.

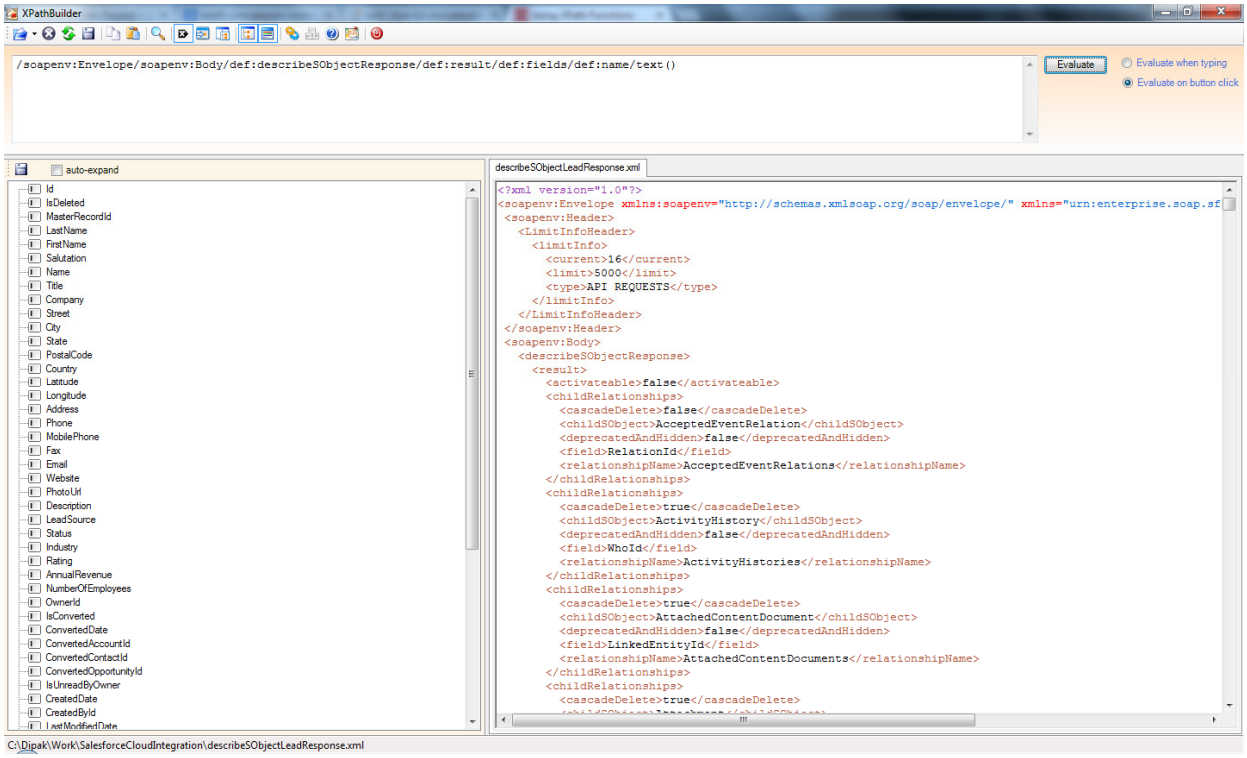

Figure-9: Screen-shot of XPathBuilder tool displaying list of API Fields for Salesforce "**Lead**" object

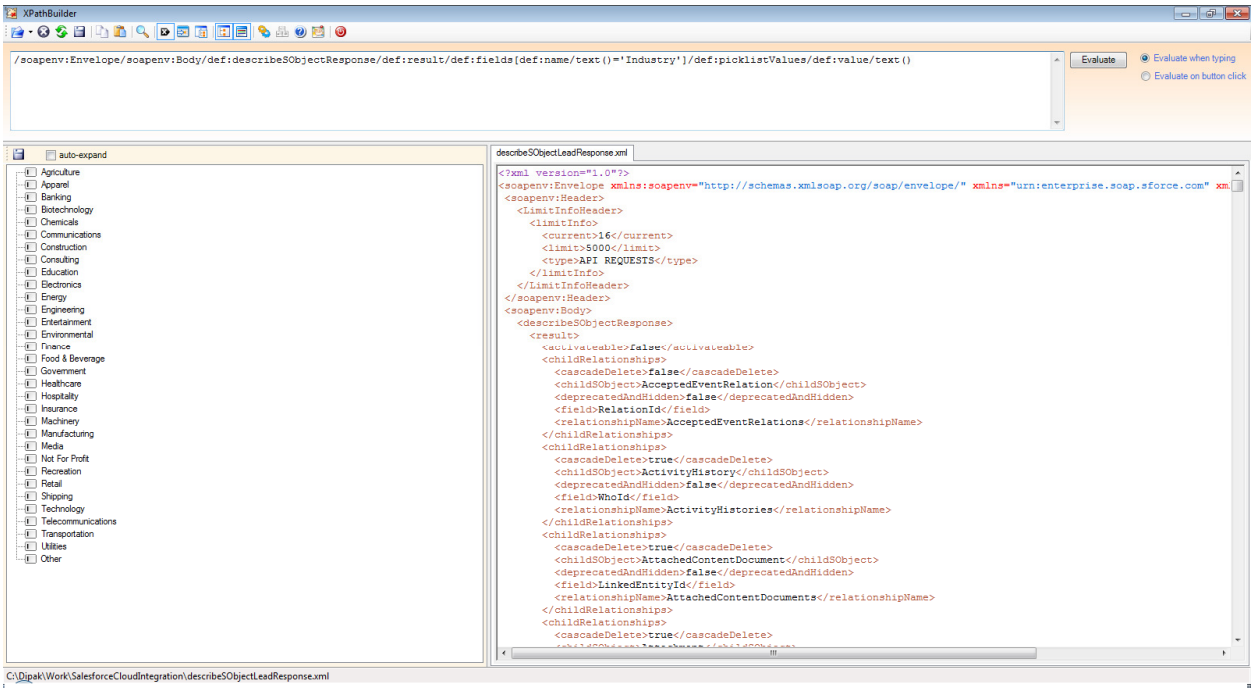

Figure-10: Screen-shot of XPathBuilder tool displaying **picklist** values of "**Industry**" field at "**Lead**" API

Once you complete the analytical study of Salesforce API fields including data types and picklist values, prepare necessary data mapping sheet(s) for each operation (e.g. createAccount, getAccountById, getContactDetails etcs.) exposed to service consumers. Note, these (createAccount, getAccountById, getContactDetails etcs.) operations are not Salesforce API operations; instead these are exposed by IBM Integration Bus to its service consumers (i.e. on-premises applications). These data mapping sheets primarily provide data transformation logic for second layer of messageflows (**i.e. Operation specific Request/Response Message Transformation Messageflows**). This article does not provide any sample data mapping sheet. Integration Architect/Business Analyst need to prepare these data mapping sheets for their organizations. However all sample XML messages (request/response) belong to the generic messageflow (**SalesforceCloudIntegration.msgflow**) are provided with this article and also illustrated at **Step-6**.

Message Type	Source	Target
Request	Request message(s) from service consumers of IBM Integration Bus (i.e. On-premises applications).	Request message(s) to the generic messageflow (**SalesforceCloudIntegration.msgflow**).
Response	Response message(s) from the generic messageflow (**SalesforceCloudIntegration.msgflow).**	Response message(s) to service consumers of IBM Integration Bus (i.e. On-premises applications).

Table-3: Data mapping Source(s) and Target(s) for second layer of messageflows

Step-5: Develop a generic messageflow for Salesforce API invocation.

This messageflow (Name: **SalesforceCloudIntegration.msgflow**) provides a MQ based request-reply service which is invoked by the second layer of messageflows (**i.e. Operation specific Request/Response Message Transformation Messageflows**). This generic messageflow is primarily responsible for Salesforce API invocation supporting some basic operations required for a simple Salesforce Cloud Integration. This article provides this messageflow as an attached artifact, you can download it and import into your Integration Bus Toolkit. **Figure-11** shows the screen-shot of IBM Integration Bus (v9) Toolkit displaying list of application artifacts of this messageflow project.

SL. No.	Salesforce API Operation
1	**C**reate
2	**R**etrieve
3	**U**pdate
4	**D**elete
5	Query

Table-4: List of supported Salesforce API operations.

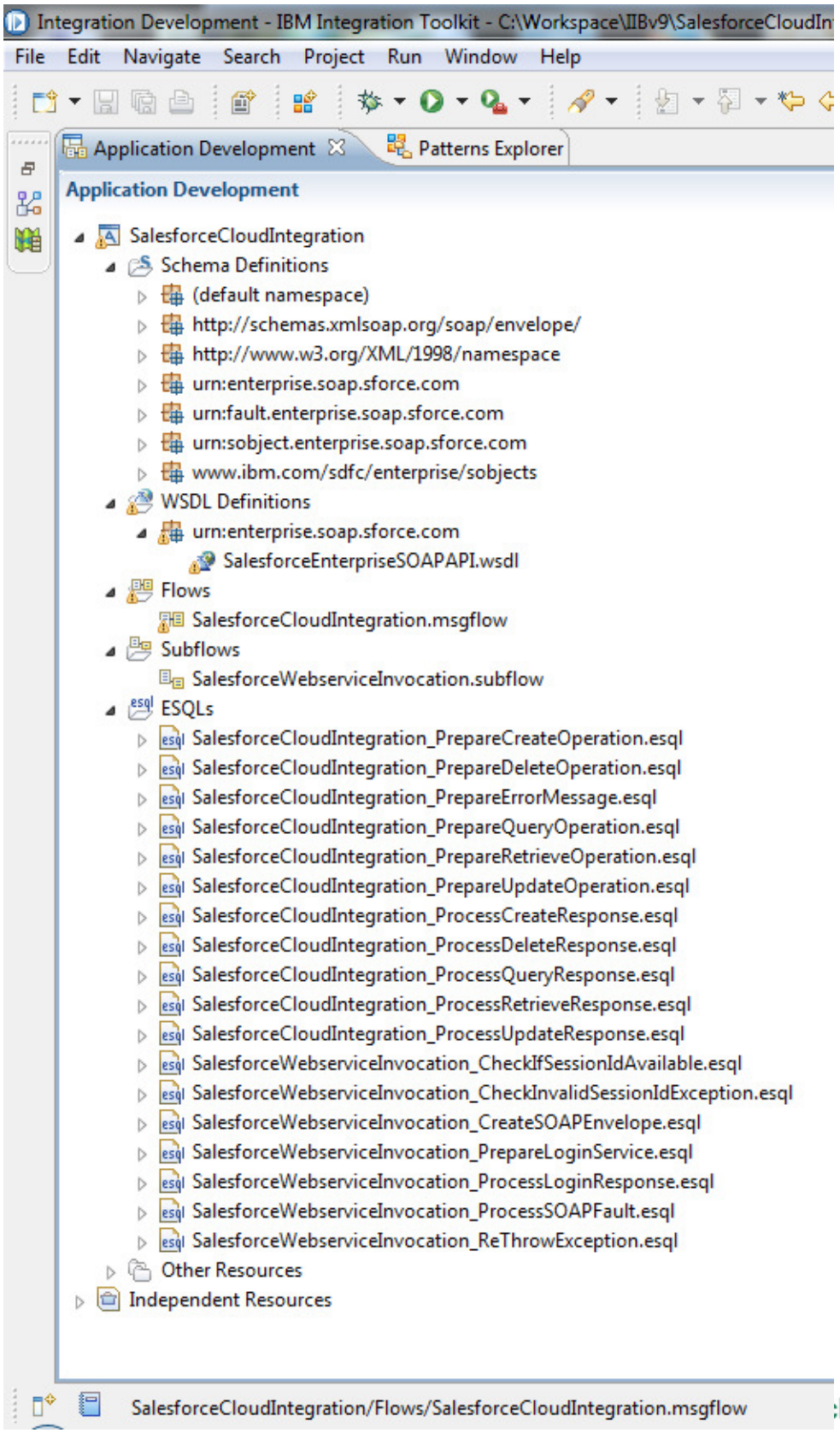
Integration Development - IBM Integration Toolkit - C:\Workspace\IIBv9\SalesforceCloudIn
File Edit Navigate Search Project Run Window Help
Application Development
Patterns Explorer
Application Development
SalesforceCloudIntegration
Schema Definitions
(default namespace)
http://schemas.xmlsoap.org/soap/envelope/
http://www.w3.org/XML/1998/namespace
urn:enterprise.soap.sforce.com
urn:fault.enterprise.soap.sforce.com
urn:sobject.enterprise.soap.sforce.com
www.ibm.com/sdfc/enterprise/sobjects
WSDL Definitions
urn:enterprise.soap.sforce.com
SalesforceEnterpriseSOAPAPI.wsdl
Flows
SalesforceCloudIntegration.msgflow
Subflows
SalesforceWebserviceInvocation.subflow
ESQLs
SalesforceCloudIntegration_PrepareCreateOperation.esql
SalesforceCloudIntegration_PrepareDeleteOperation.esql
SalesforceCloudIntegration_PrepareErrorMessage.esql
SalesforceCloudIntegration_PrepareQueryOperation.esql
SalesforceCloudIntegration_PrepareRetrieveOperation.esql
SalesforceCloudIntegration_PrepareUpdateOperation.esql
SalesforceCloudIntegration_ProcessCreateResponse.esql
SalesforceCloudIntegration_ProcessDeleteResponse.esql
SalesforceCloudIntegration_ProcessQueryResponse.esql
SalesforceCloudIntegration_ProcessRetrieveResponse.esql
SalesforceCloudIntegration_ProcessUpdateResponse.esql
SalesforceWebserviceInvocation_CheckIfSessionIdAvailable.esql
SalesforceWebserviceInvocation_CheckInvalidSessionIdException.esql
SalesforceWebserviceInvocation_CreateSOAPEnvelope.esql
SalesforceWebserviceInvocation_PrepareLoginService.esql
SalesforceWebserviceInvocation_ProcessLoginResponse.esql
SalesforceWebserviceInvocation_ProcessSOAPFault.esql
SalesforceWebserviceInvocation_ReThrowException.esql
Other Resources
Independent Resources
SalesforceCloudIntegration/Flows/SalesforceCloudIntegration.msgflow

Figure-11: IBM Integration Bus (v9) Application Artifacts for Salesforce Cloud Integration

Step-5.1: Create a Message Model from Salesforce Enterprise API WSDL

Create a Message Model at Integration Bus Toolkit, use the Salesforce Enterprise API WSDL downloaded at **Step-3**.

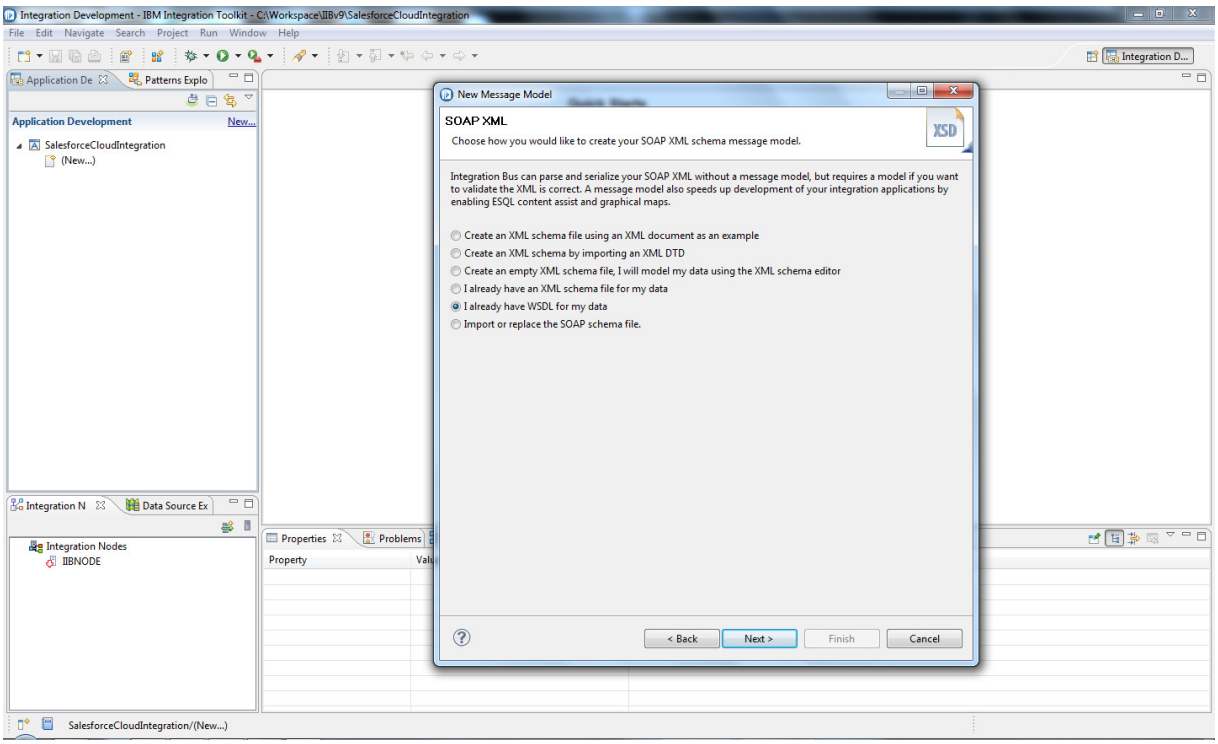

Figure-12: Message Model creation (1) from Salesforce Enterprise API WSDL

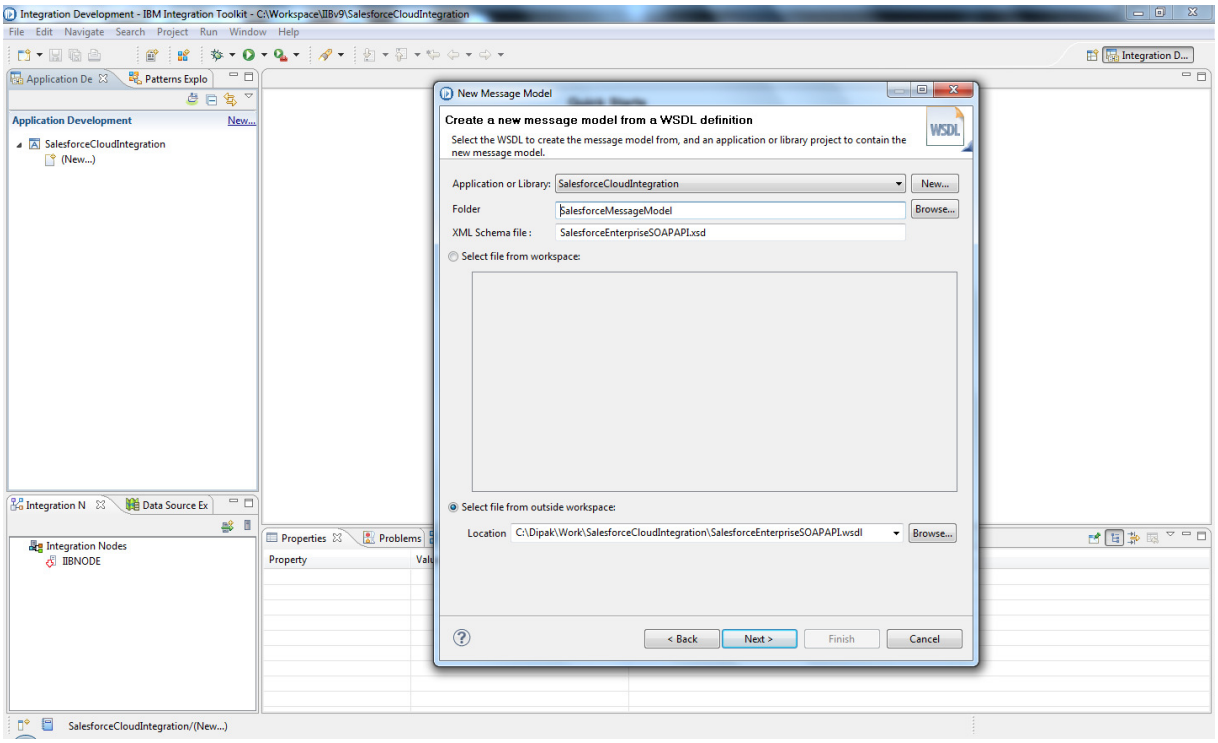

Figure-13: Message Model creation (2) from Salesforce Enterprise API WSDL

Step-5.2: Develop a subflow implementing Salesforce API invocation

Develop a subflow (name: **SalesforceWebserviceInvocation.subflow**) implementing Salesforce API invocation including "**login**" operation. Refer to **Figure-14** representing messageflow diagram of this subflow. You don't need to develop this subflow from scratch, you can just download the attached messageflow project and import into Toolkit. This subflow implements following activities.

1. First checks whether existing "**SessionId**" and "**ServerUrl**" are available from previous invocation. If available, it skips the invocation of "**login**" operation, otherwise it prepares "**login**" request message (SOAP) and invokes the Salesforce "**login**" operation. Upon successful invocation of "**login**" operation, it caches "**SessionId**" and "**ServerUrl**" into shared variables for future invocation of the messageflow.
2. After getting a valid Salesforce "**SessionId**" and "**ServerUrl**", this subflow creates a SOAP Envelope with necessary SOAP Header containing the "**SessionId**", and wraps the request XML message with this SOAP Envelope.
3. Invokes the specific Salesforce API operation (e.g. create, retrieve, update, delete, query) and then extracts the SOAP envelope from API response message upon successful invocation.
4. In case of SOAP Fault returned by Salesforce API invocation, it throws a User Exception after extracting the error code and error message from SOAP Fault.
5. All the above mentioned activities are enclosed within a Try-Catch node to implement retry mechanism in case of "**INVALID_SESSION_ID**" error thrown by Salesforce API invocation. If the

Salesforce login session remains idle beyond a predefined period of time, the session becomes invalid and it requires re-invocation of "**login**" operation prior to the next Salesforce API call.

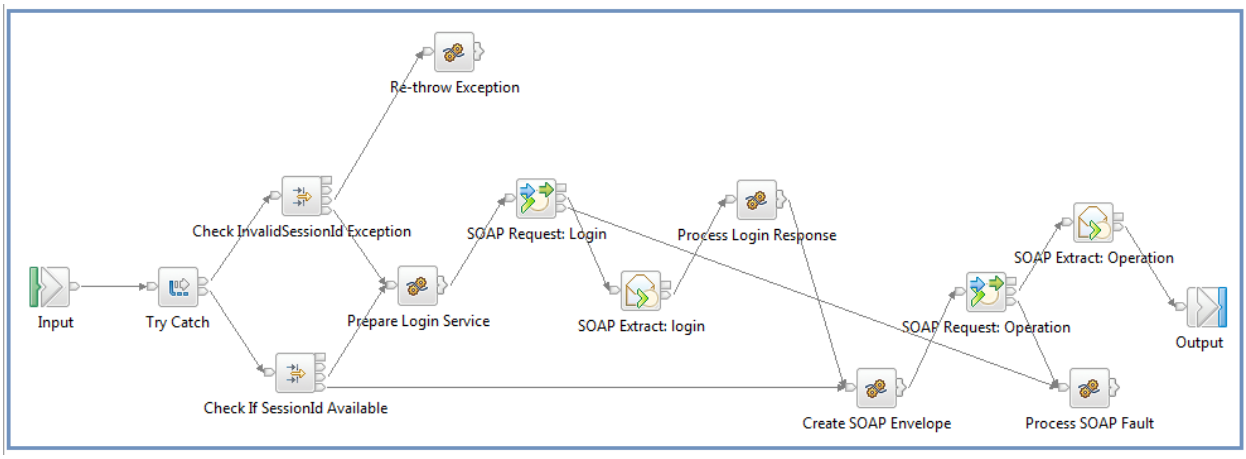

Figure-14: Messageflow diagram for **SalesforceWebserviceInvocation.subflow**

Figure-15 represents a screen-shot of User Defined Parameters (promoted to flow level) of this subflow. Make sure you set appropriate UDP values prior to deployment.

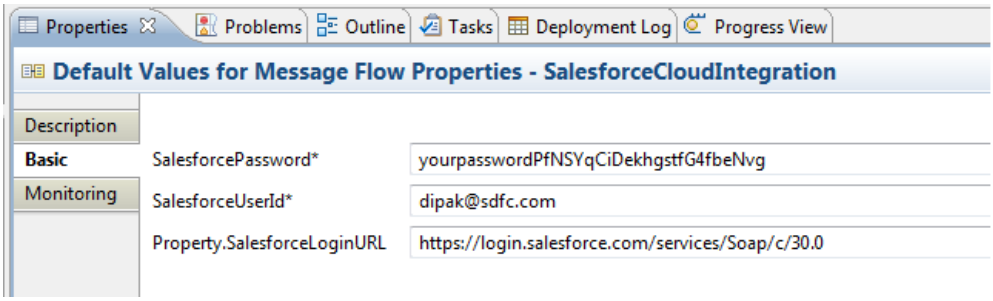

Figure-15: User Defined Parameters (promoted to flow level) used by the subflow

Step-5.3: Develop a common messageflow

Develop a common messageflow (name: **SalesforceCloudIntegration.msgflow**) implementing MQ based request-reply service for Salesforce Cloud Integration. Refer to **Figure-16** for messageflow diagram, and messageflow code attached with this article for low level coding logic. You don't need to develop this messageflow from scratch, you can just download the attached messageflow project and import into your Integration Bus Toolkit. Increase number of additional instances while deploying the messageflow to avoid performance bottleneck. Following activities are implemented within this messageflow.

1. Receives generic XML message for all supported operations (Create/Retrieve/Update/Delete/Query) through MQ Input node. Sample messages for all supported operations are attached as additional artifacts with this article. Refer to **Step-6** for request/response XML structure.

2. Based on the "**operation**" specified in message body, a **Route** node routes the input message to corresponding compute node for preparing Salesforce API operation(s) specific XML message. Refer to underlying ESQL code for detail implementation logic.
3. It then invokes the "**SalesforceWebserviceInvocation**" subflow which performs actual Salesforce API invocation including Salesforce login and session management.
4. Upon successful Salesforce API invocation, another **Route** node again routes the Salesforce API response message to corresponding compute node for preparation of final response message. Refer to underlying ESQL code for detail implementation logic.
5. Put the final response message into MQ reply queue.
6. In case of error, this messageflow prepares a generic error message and put into the reply queue. However if required you can implement error handling mechanism and transaction monitoring system as per your organization standard.

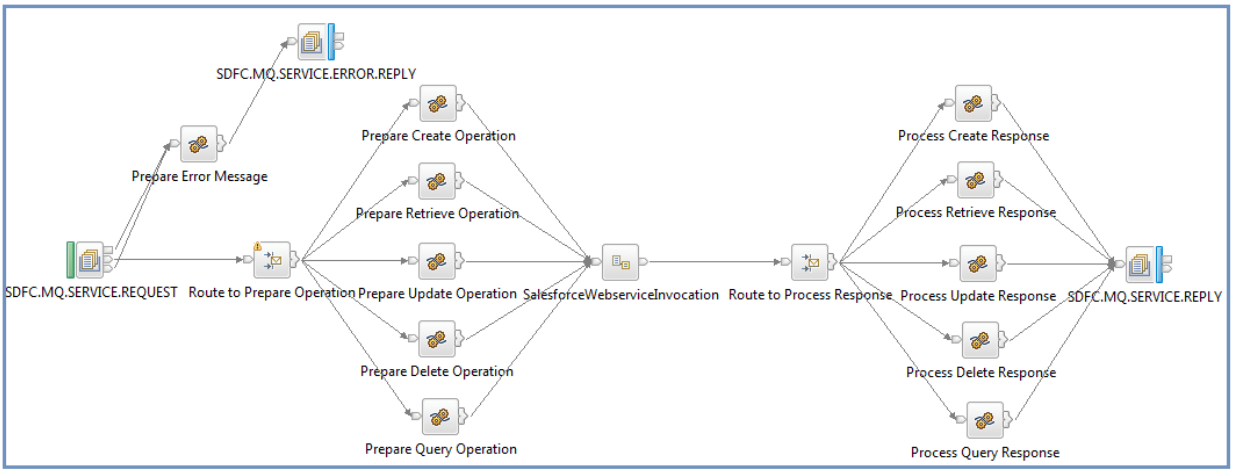

Figure-16: Flow diagram for **SalesforceCloudIntegration.msgflow**

Attached messageflow project uses Salesforce Enterprise API WSDL in Message Model and corresponding namespaces in ESQL code for preparing/processing Salesforce Webservice SOAP messages. However you can easily switch to Salesforce Partner API integration by using a Partner WSDL in Message Model and corresponding namespaces in ESQL code.

Namespace declaration in ESQL code (SalesforceWebserviceInvocation_PrepareLoginService.esql) used for Salesforce Enterprise API Invocation:

```
DECLARE urn NAMESPACE 'urn:enterprise.soap.sforce.com';
DECLARE urn1 NAMESPACE 'urn:sobject.enterprise.soap.sforce.com';
DECLARE sf NAMESPACE 'urn:fault.enterprise.soap.sforce.com';
```

Namespace declaration required for Salesforce Partner API Invocation:

```
DECLARE urn NAMESPACE 'urn:partner.soap.sforce.com';
```

DECLARE urn1 NAMESPACE 'urn:sobject.partner.soap.sforce.com';
DECLARE sf NAMESPACE 'urn:fault.partner.soap.sforce.com';

Step-6: Preparation of test messages and testing of the messageflow

This article uses Salesforce "**Lead**" Object for all test scenarios. However all supported operations can be performed on all other Salesforce objects in similar fashions. Following sub-steps demonstrate testing of each supported operation including sample request/response/error XMLs.

Step-6.1: Normal Test Scenario: "Create" operation

Table-5 represents a sample XML message for "**Create**" operation on Salesforce "**Lead**" object. Multiple records can be created in Salesforce through a single invocation of this messageflow. Element names and sequences under <**Record**> node are exactly same as Salesforce API fields in "**Lead**" object analyzed at **Step-4** through SOAPUI tool. This article uses RFHUtil tool to put request XML into input queue of the **SalesforceCloudIntegration messageflow**. **Figure-17** represents a screen-shot of Salesforce UI displaying newly created "**Lead**" records after successful invocation of this messageflow. **Table-6** shows the corresponding response message captured from reply queue of this messageflow, containing execution status and Salesforce record **Id** for each "**Lead**" record. This Salesforce record Id will be used for Retrieve, Update and Delete operations as key field.

```
<?xml version="1.0" encoding="UTF-8"?>
<Salesforce>
  <Object type="Lead" operation="create">
    <Records>
      <Record>
        <Salutation>Mr.</Salutation>
        <FirstName>Dipak</FirstName>
        <LastName>Pal</LastName>
        <Title>Software Engineer</Title>
        <Company>IBM</Company>
        <Street>11 New Orchard Road</Street>
        <City>Armonk</City>
        <State>New York</State>
        <PostalCode>10504</PostalCode>
        <Country>United States</Country>
        <MobilePhone>111-222-1234</MobilePhone>
        <Email>email1@ibm.com</Email>
        <Status>Working - Contacted</Status>
        <Industry>Agriculture</Industry>
        <Rating>Cold</Rating>
        <AnnualRevenue>10000000000</AnnualRevenue>
        <NumberOfEmployees>300000</NumberOfEmployees>
        <SICCode__c>123</SICCode__c>
        <ProductInterest__c>GC5000 series</ProductInterest__c>
        <Primary__c>Yes</Primary__c>
        <CurrentGenerators__c>All</CurrentGenerators__c>
        <NumberofLocations__c>100</NumberofLocations__c>
      </Record>
      <Record>
        <Salutation>Mr.</Salutation>
        <FirstName>Jow</FirstName>
        <LastName>Brown</LastName>
        <Title>Software Engineer</Title>
        <Company>IBM</Company>
```

```
        <Street>12 New Orchard Road</Street>
        <City>Armonk</City>
        <State>New York</State>
        <PostalCode>10504</PostalCode>
        <Country>United States</Country>
        <MobilePhone>333-222-1234</MobilePhone>
        <Email>email2@ibm.com</Email>
        <Status>Working - Contacted</Status>
        <Industry>Agriculture</Industry>
        <Rating>Cold</Rating>
        <AnnualRevenue>10000000000</AnnualRevenue>
        <NumberOfEmployees>300000</NumberOfEmployees>
        <SICCode__c>123</SICCode__c>
        <ProductInterest__c>GC5000 series</ProductInterest__c>
        <Primary__c>Yes</Primary__c>
        <CurrentGenerators__c>All</CurrentGenerators__c>
        <NumberofLocations__c>100</NumberofLocations__c>
      </Record>
      <Record>
        <Salutation>Mr.</Salutation>
        <FirstName>Hari</FirstName>
        <LastName>Prasad</LastName>
        <Title>Software Engineer</Title>
        <Company>IBM</Company>
        <Street>13 New Orchard Road</Street>
        <City>Armonk</City>
        <State>New York</State>
        <PostalCode>10504</PostalCode>
        <Country>United States</Country>
        <MobilePhone>999-222-1234</MobilePhone>
        <Email>email3@ibm.com</Email>
        <Status>Working - Contacted</Status>
        <Industry>Agriculture</Industry>
        <Rating>Cold</Rating>
        <AnnualRevenue>10000000000</AnnualRevenue>
        <NumberOfEmployees>300000</NumberOfEmployees>
        <SICCode__c>123</SICCode__c>
        <ProductInterest__c>GC5000 series</ProductInterest__c>
        <Primary__c>Yes</Primary__c>
        <CurrentGenerators__c>All</CurrentGenerators__c>
        <NumberofLocations__c>100</NumberofLocations__c>
      </Record>
    </Records>
  </Object>
</Salesforce>
```

Table-5: Sample request XML for "**Create**" operation

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<Salesforce>
  <Object type="Lead" operation="create">
    <Records>
      <Record>
        <Status>Success</Status>
        <Id>00QF000000btnoTMAQ</Id>
      </Record>
      <Record>
        <Status>Success</Status>
        <Id>00QF000000btnoUMAQ</Id>
      </Record>
      <Record>
        <Status>Success</Status>
        <Id>00QF000000btnoVMAQ</Id>
      </Record>
    </Records>
```

```
  </Object>
</Salesforce>
```

Table-6: Sample response XML for "**Create**" operation

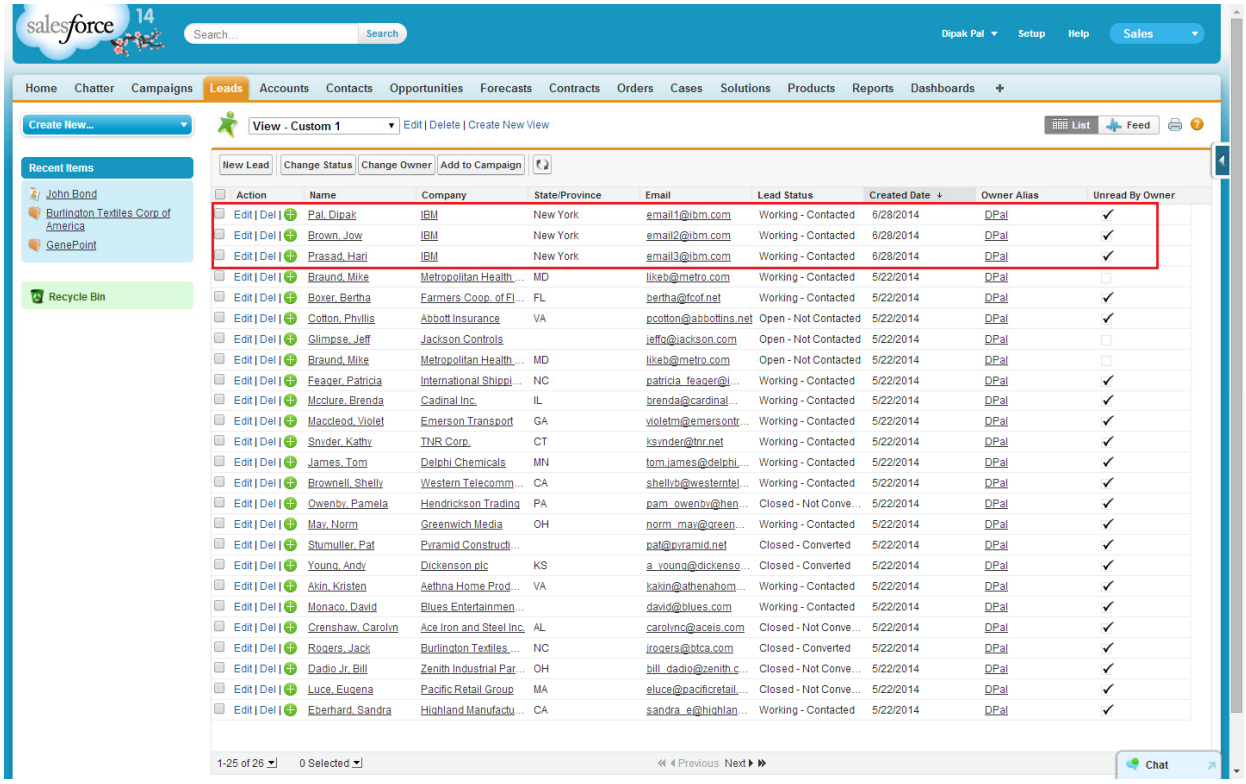

Figure-17: Screen-shot of Salesforce UI displaying newly created "**Lead**" records

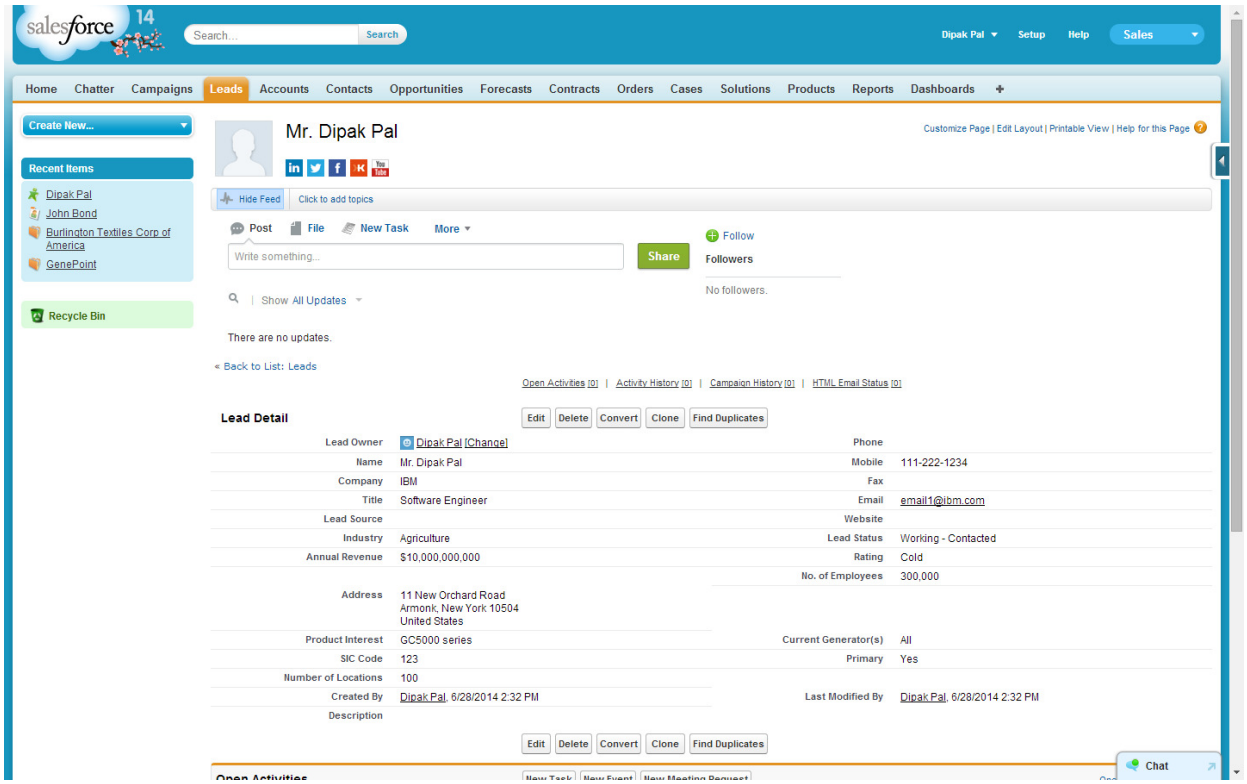

Figure-18: Salesforce UI screen-shot displaying a detail "**Lead**" record created by "**Create**" operation

Step-6.2: Error Test Scenario: "Create" operation

Create a request XML containing an invalid element which does not exist in Salesforce API field listing. **Table-7** represents a request XML which contains an invalid element (<**InvalidField**>). Error response XML replied by this messageflow is shown at **Table-8**. Messageflow itself does not validate the request XML, it invokes the Salesforce API ("**create**" operation) which returns a SOAP Fault and the messageflow prepares error response based on this SOAP Fault.

```
<?xml version="1.0" encoding="UTF-8"?>
<Salesforce>
  <Object type="Lead" operation="create">
    <Records>
      <Record>
        <InvalidField>InvalidField</InvalidField>
        <Salutation>Mr.</Salutation>
        <FirstName>Dipak</FirstName>
        <LastName>Pal</LastName>
        <Title>Software Engineer</Title>
        <Company>IBM</Company>
        <Street>11 New Orchard Road</Street>
        <City>Armonk</City>
        <State>New York</State>
        <PostalCode>10504</PostalCode>
        <Country>United States</Country>
        <MobilePhone>111-222-1234</MobilePhone>
        <Email>email1@ibm.com</Email>
        <Status>Working - Contacted</Status>
```

```
        <Industry>Agriculture</Industry>
        <Rating>Cold</Rating>
        <AnnualRevenue>10000000000</AnnualRevenue>
        <NumberOfEmployees>300000</NumberOfEmployees>
        <SICCode__c>123</SICCode__c>
        <ProductInterest__c>GC5000 series</ProductInterest__c>
        <Primary__c>Yes</Primary__c>
        <CurrentGenerators__c>All</CurrentGenerators__c>
        <NumberofLocations__c>100</NumberofLocations__c>
      </Record>
    </Records>
  </Object>
</Salesforce>
```

Table-7: A sample request XML containing an invalid field (<**InvalidField**>)

```
<?xml version="1.0" encoding="UTF-8" standalone="yes" ?>
<Salesforce>
  <Error>
    <ErrorCode>SDFC_ERROR</ErrorCode>
    <ErrorMessage>User generated exception: User generated exception: INVALID_FIELD: No such column 'InvalidField' on entity 'Lead'. If you are attempting to use a custom field, be sure to append the '__c' after the custom field name. Please reference your WSDL or the describe call for the appropriate names.</ErrorMessage>
  </Error>
</Salesforce>
```

Table-8: A sample error response XML due to an invalid field in request XML

Step-6.3: Normal Test Scenario: "Retrieve" operation

Table-9 contains a sample XML message for "**Retrieve**" operation on Salesforce "**Lead**" object. Multiple records can be retrieved from Salesforce through a single invocation of this messageflow. This operation requires record Id(s) and a list of Salesforce API field name(s) as input parameters. Refer to **Step-4** for obtaining Salesforce API field name(s). This test scenario retrieves the same set of records created during "**Create**" operation discussed at **Step-6.1**. **Table-10** contains the corresponding response message. **Table-11** and **Table-12** describe another set of request-response messages of "**Retrieve**" operation on Salesforce "**Contact**" object.

```
<?xml version="1.0" encoding="UTF-8"?>
<Salesforce>
  <Object type="Lead" operation="retrieve">
    <Records>
      <Record>
        <Id>00QF000000btnoTMAQ</Id>
      </Record>
      <Record>
        <Id>00QF000000btnoUMAQ</Id>
      </Record>
      <Record>
        <Id>00QF000000btnoVMAQ</Id>
      </Record>
    </Records>
    <Fields>
      <Field>Id</Field>
      <Field>Salutation</Field>
      <Field>FirstName</Field>
      <Field>LastName</Field>
      <Field>Company</Field>
    </Fields>
  </Object>
```

```
</Salesforce>
```

Table-9: Sample request XML for “Retrieve” operation on Salesforce “Lead” object

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<Salesforce>
  <Object type="Lead" operation="retrieve">
    <Records>
      <Record>
        <Id>00QF000000btnoTMAQ</Id>
        <Company>Microsoft</Company>
        <FirstName>Dipak</FirstName>
        <LastName>Pal</LastName>
        <Salutation>Mr.</Salutation>
      </Record>
      <Record>
        <Id>00QF000000btnoUMAQ</Id>
        <Company>Microsoft</Company>
        <FirstName>Jow</FirstName>
        <LastName>Brown</LastName>
        <Salutation>Mr.</Salutation>
      </Record>
      <Record>
        <Id>00QF000000btnoVMAQ</Id>
        <Company>IBM</Company>
        <FirstName>Hari</FirstName>
        <LastName>Prasad</LastName>
        <Salutation>Mr.</Salutation>
      </Record>
    </Records>
  </Object>
</Salesforce>
```

Table-10: Sample response XML for “Retrieve” operation on Salesforce “Lead” object

```
<?xml version="1.0" encoding="UTF-8"?>
<Salesforce>
  <Object type="Contact" operation="retrieve">
    <Records>
      <Record>
        <Id>003F000001SBki3IAD</Id>
      </Record>
      <Record>
        <Id>00QF000000bsB2gMAE</Id>
      </Record>
      <Record>
        <Id>003F000001SBki5IAD</Id>
      </Record>
    </Records>
    <Fields>
      <Field>Title</Field>
      <Field>Name</Field>
      <Field>Account.Id</Field>
      <Field>Account.Name</Field>
    </Fields>
  </Object>
</Salesforce>
```

Table-11: Sample request XML for “Retrieve” operation on Salesforce “Contact” object

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<Salesforce>
  <Object type="Contact" operation="retrieve">
    <Records>
      <Record>
```

```
        <Id>003F000001SBki3IAD</Id>
        <Account>
          <Id>001F00000175gzsIAA</Id>
          <Name>Edge Communications</Name>
        </Account>
        <Name>Rose Gonzalez</Name>
        <Title>SVP, Procurement</Title>
      </Record>
      <Record/>
      <Record>
        <Id>003F000001SBki5IAD</Id>
        <Account>
          <Id>001F00000175gztIAA</Id>
          <Name>Burlington Textiles Corp of America</Name>
        </Account>
        <Name>Jack Rogers</Name>
        <Title>VP, Facilities</Title>
      </Record>
    </Records>
  </Object>
</Salesforce>
```

Table-12: Sample response XML for "Retrieve" operation on Salesforce "Contact" object

Step-6.4: Error Test Scenario: "Retrieve" operation

Create a request XML containing an invalid record Id which does not exist in Salesforce. **Table-13** represents a request XML which contains an invalid Id (e.g. <Id>**00QF000000btnoXXXX**</Id>). Corresponding error response XML is shown at **Table-14**. Messageflow itself does not validate the request XML, it invokes the Salesforce API ("**retrieve**" operation) which returns a SOAP Fault and the messageflow prepares error response based on this SOAP Fault.

```
<?xml version="1.0" encoding="UTF-8"?>
<Salesforce>
  <Object type="Lead" operation="retrieve">
    <Records>
      <Record>
        <Id>00QF000000btnoTMAQ</Id>
      </Record>
      <Record>
        <Id>00QF000000btnoXXXX</Id>
      </Record>
      <Record>
        <Id>00QF000000btnoVMAQ</Id>
      </Record>
    </Records>
    <Fields>
      <Field>Id</Field>
      <Field>Salutation</Field>
      <Field>FirstName</Field>
      <Field>LastName</Field>
      <Field>Company</Field>
    </Fields>
  </Object>
</Salesforce>
```

Table-13: A sample request XML containing an invalid record Id

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<Salesforce>
  <Error>
```

```
    <ErrorCode>SDFC_ERROR</ErrorCode>
    <ErrorMessage>User generated exception: MALFORMED_ID: malformed id 00QF000000btnoXXXX</ErrorMessage>
  </Error>
</Salesforce>
```

Table-14: A sample error response XML due to an invalid record Id in request XML

Step-6.5: Test Scenario: "Update" operation

Table-15 represents a sample request XML for "**Update**" operation on Salesforce "**Lead**" object. Multiple records can be updated in Salesforce through a single invocation of this messageflow. This operation requires record Id(s) including list of Salesforce API field name(s) and corresponding value(s) as input parameters. Prior to execute this test scenario, a record (Id: **00QF000000btnoVMAQ**) has been deleted manually from "**Lead**" table through Salesforce UI such that an error test case can be included into a single test scenario. Execution of this test scenario updates the records created at previously described "**Create**" operation (**Step-6.1**). **Table-16** contains the corresponding response XML replied by the messageflow, where update operations on first two records are successful but failed for third record (Id: **00QF000000btnoVMAQ**) as the record is already deleted.

```
<?xml version="1.0" encoding="UTF-8"?>
<Salesforce>
  <Object type="Lead" operation="update">
    <Records>
      <Record>
        <Id>00QF000000btnoTMAQ</Id>
        <Title>Software Engineer</Title>
        <Company>Microsoft</Company>
        <Email>email1@microsoft.com</Email>
        <Status>Working - Contacted</Status>
      </Record>
      <Record>
        <Id>00QF000000btnoUMAQ</Id>
        <Title>Software Engineer</Title>
        <Company>Microsoft</Company>
        <Street>12 New Orchard Road</Street>
        <City>Armonk</City>
        <State>New York</State>
        <PostalCode>10504</PostalCode>
        <Country>United States</Country>
        <MobilePhone>333-222-1234</MobilePhone>
        <Email>email2@microsoft.com</Email>
        <Status>Working - Contacted</Status>
      </Record>
      <Record>
        <Id>00QF000000btnoVMAQ </Id>
        <Title>Software Engineer</Title>
        <Company>Microsoft</Company>
        <Street>13 New Orchard Road</Street>
        <City>Armonk</City>
        <State>New York</State>
        <PostalCode>10504</PostalCode>
        <Country>United States</Country>
        <MobilePhone>999-222-1234</MobilePhone>
        <Email>email3@microsoft.com</Email>
        <Status>Working - Contacted</Status>
      </Record>
    </Records>
  </Object>
</Salesforce>
```

Table-15: Sample request XML for "**Update**" operation on Salesforce "**Lead**" object

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<Salesforce>
  <Object type="Lead" operation="update">
    <Records>
      <Record>
        <Status>Success</Status>
        <Id>00QF000000btnoTMAQ</Id>
      </Record>
      <Record>
        <Status>Success</Status>
        <Id>00QF000000btnoUMAQ</Id>
      </Record>
      <Record>
        <Status>Error</Status>
        <Error>
          <ErrorCode>ENTITY_IS_DELETED</ErrorCode>
          <ErrorMessage>entity is deleted</ErrorMessage>
        </Error>
      </Record>
    </Records>
  </Object>
</Salesforce>
```

Table-16: A Sample response XML for "**Update**" operation on Salesforce "**Lead**" object

Step-6.6: Test Scenario: "Delete" operation

Table-17 shows a sample request XML for "**Delete**" operation on Salesforce "**Lead**" object. Multiple records can be deleted from Salesforce through a single invocation of this messageflow. This operation requires only record Id(s) as input parameter. Execution of this test scenario deletes the records created at previously described "**Create**" operation (**Step-6.1**). This test scenario includes an Invalid record Id (**00QF000000btnoUXXX**) which does not exist in Salesforce such that an error test case can be added into the same test scenario. **Table-18** represents the corresponding response XML replied by the messageflow, where delete operations on first two records are successful but failed for third record (Id: **00QF000000btnoUXXX**) as the record does not exist in Salesforce. **Table-19** shows another response XML when the same request XML (**Table-17**) is put into the input queue of the messageflow for second time. **Figure-19** represents the Salesforce UI screen-shot of deleted records upon successful delete operation.

```
<?xml version="1.0" encoding="UTF-8"?>
<Salesforce>
  <Object type="Lead" operation="delete">
    <Records>
      <Record>
        <Id>00QF000000btnoTMAQ</Id>
      </Record>
      <Record>
        <Id>00QF000000btnoUMAQ</Id>
      </Record>
      <Record>
        <Id>00QF000000btnoUXXX</Id>
      </Record>
    </Records>
  </Object>
```

```
</Salesforce>
```

Table-17: Sample request XML for "**Delete**" operation on Salesforce "**Lead**" object.

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<Salesforce>
  <Object type="Lead" operation="delete">
    <Records>
      <Record>
        <Status>Success</Status>
        <Id>00QF000000btnoTMAQ</Id>
      </Record>
      <Record>
        <Status>Success</Status>
        <Id>00QF000000btnoUMAQ</Id>
      </Record>
      <Record>
        <Status>Error</Status>
        <Error>
          <ErrorCode>MALFORMED_ID</ErrorCode>
          <ErrorMessage>malformed id 00QF000000btnoUXXX</ErrorMessage>
        </Error>
      </Record>
    </Records>
  </Object>
</Salesforce>
```

Table-18: Sample response XML for "**Delete**" operation on Salesforce "**Lead**" object.

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<Salesforce>
  <Object type="Lead" operation="delete">
    <Records>
      <Record>
        <Status>Error</Status>
        <Error>
          <ErrorCode>ENTITY_IS_DELETED</ErrorCode>
          <ErrorMessage>entity is deleted</ErrorMessage>
        </Error>
      </Record>
      <Record>
        <Status>Error</Status>
        <Error>
          <ErrorCode>ENTITY_IS_DELETED</ErrorCode>
          <ErrorMessage>entity is deleted</ErrorMessage>
        </Error>
      </Record>
      <Record>
        <Status>Error</Status>
        <Error>
          <ErrorCode>MALFORMED_ID</ErrorCode>
          <ErrorMessage>malformed id 00QF000000btnoUXXX</ErrorMessage>
        </Error>
      </Record>
    </Records>
  </Object>
</Salesforce>
```

Table-19: Sample response XML for "**Delete**" operation on Salesforce "**Lead**" object

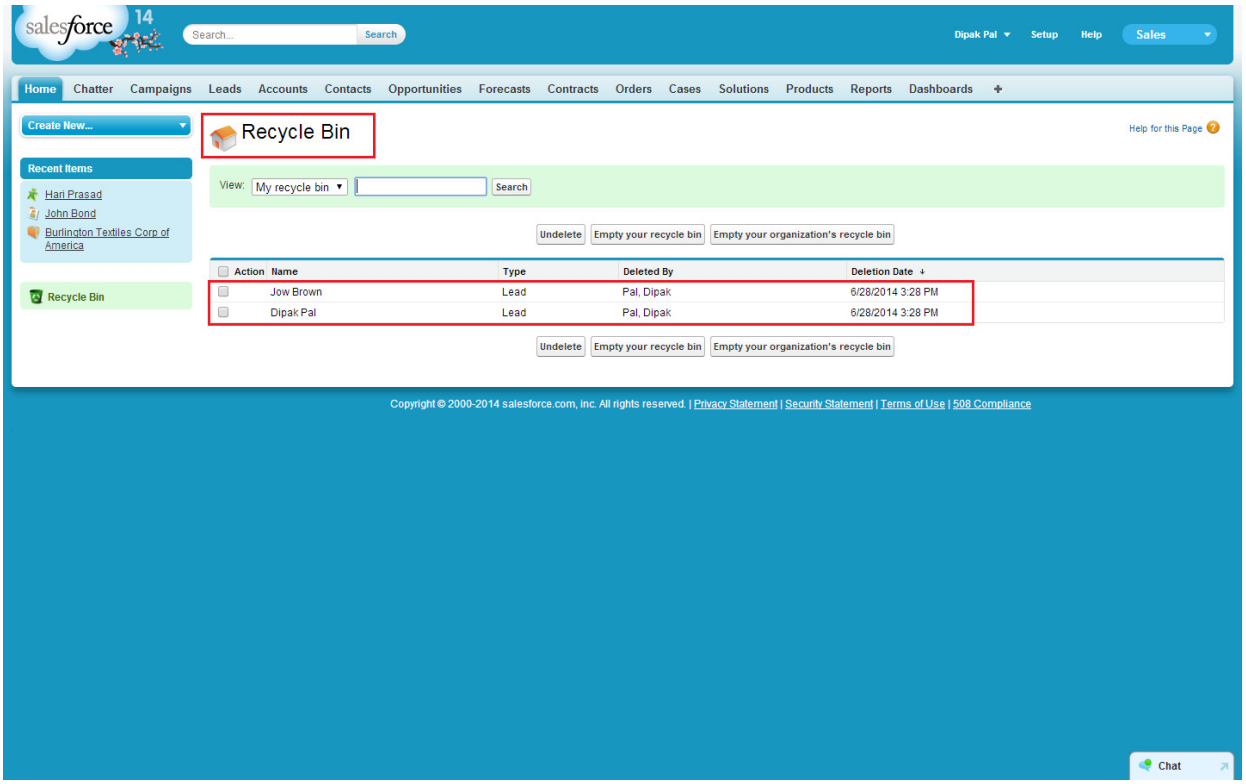

Figure-19: Screen-shot of Salesforce UI displaying deleted records in **Recycle Bin**

Step-6.7: Normal Test Scenario: "Query" operation

Table-20 contains a sample request XML for "**Query**" operation on Salesforce "**Lead**" object. Only one **SOQL** query can be executed through a single invocation of this messageflow. Discussion of **SOQL** query development is beyond the scope of this article. Refer to ***Force.com SOQL and SOSL Reference*** (http://www.salesforce.com/us/developer/docs/soql_sosl/) for SOQL query development. **Table-21** contains the corresponding response XML consisting of formatted result of the SOQL query execution. **Table-22** and **Table-23** describe another set of request-response messages of "**Query**" operation on Salesforce "**Contact**" object.

```
<?xml version="1.0" encoding="UTF-8"?>
<Salesforce>
  <Object operation="query">
    <Query>select Id, Salutation, FirstName, LastName, Company, Street, City, State, PostalCode, Country from Lead where Company =
'IBM'</Query>
  </Object>
</Salesforce>
  </Object>
</Salesforce>
```

Table-20: Sample request XML for "**Query**" operation on Salesforce "**Lead**" object.

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<Salesforce>
  <Object operation="query">
```

```
    <Records count="3">
      <Record>
        <Id>00QF000000btnoTMAQ</Id>
        <City>Armonk</City>
        <Company>IBM</Company>
        <Country>United States</Country>
        <FirstName>Dipak</FirstName>
        <LastName>Pal</LastName>
        <PostalCode>10504</PostalCode>
        <Salutation>Mr.</Salutation>
        <State>New York</State>
        <Street>11 New Orchard Road</Street>
      </Record>
      <Record>
        <Id>00QF000000btnoUMAQ</Id>
        <City>Armonk</City>
        <Company>IBM</Company>
        <Country>United States</Country>
        <FirstName>Jow</FirstName>
        <LastName>Brown</LastName>
        <PostalCode>10504</PostalCode>
        <Salutation>Mr.</Salutation>
        <State>New York</State>
        <Street>12 New Orchard Road</Street>
      </Record>
      <Record>
        <Id>00QF000000btnoVMAQ</Id>
        <City>Armonk</City>
        <Company>IBM</Company>
        <Country>United States</Country>
        <FirstName>Hari</FirstName>
        <LastName>Prasad</LastName>
        <PostalCode>10504</PostalCode>
        <Salutation>Mr.</Salutation>
        <State>New York</State>
        <Street>13 New Orchard Road</Street>
      </Record>
    </Records>
  </Object>
</Salesforce>
```

Table-21: Sample response XML for "**Query**" operation on Salesforce "**Lead**" object

```
<?xml version="1.0" encoding="UTF-8"?>
<Salesforce>
  <Object operation="query">
    <Query>select Id, Title, Name, Account.Id, Account.Name from Contact</Query>
  </Object>
</Salesforce>
```

Table-22: Sample request XML for "**Query**" operation on Salesforce "**Contact**" object

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<Salesforce>
  <Object operation="query">
    <Records count="20">
      <Record>
        <Id>003F000001SBki3IAD</Id>
        <Account>
          <Id>001F00000175gzsIAA</Id>
          <Name>Edge Communications</Name>
        </Account>
        <Name>Rose Gonzalez</Name>
        <Title>SVP, Procurement</Title>
      </Record>
```

```
        <Record>
          <Id>003F000001SBki4IAD</Id>
          <Account>
            <Id>001F00000175gzsIAA</Id>
            <Name>Edge Communications</Name>
          </Account>
          <Name>Sean Forbes</Name>
          <Title>CFO</Title>
        </Record>
      </Records>
    </Object>
</Salesforce>
```

Table-23: Sample response XML for "**Query**" operation on Salesforce "**Contact**" object

Step-6.8: Error Test Scenario: "Query" operation

Create a request XML containing an invalid field name in SOQL query, which does not belong to Salesforce API field list. **Table-24** represents a request XML which contains an invalid field name (e.g. **InvalidField**) in SOQL query. Corresponding error response XML is shown at **Table-25**. Messageflow itself does not validate the SOQL query, it invokes the Salesforce API ("**query**" operation) which returns a SOAP Fault and the messageflow prepares error response based on this SOAP Fault.

```
<?xml version="1.0" encoding="UTF-8"?>
<Salesforce>
  <Object operation="query">
    <Query>select Id, InvalidField from Lead</Query>
  </Object>
</Salesforce>
```

Table-24: A sample request XML containing an invalid field in SOQL query.

```
<?xml version="1.0" encoding="UTF-8" standalone="yes" ?>
<Salesforce>
  <Error>
    <ErrorCode>SDFC_ERROR</ErrorCode>
    <ErrorMessage>User generated exception: User generated exception: INVALID_FIELD: select Id, InvalidField from Lead ^ ERROR at
Row:1:Column:12 No such column 'InvalidField' on entity 'Lead'. If you are attempting to use a custom field, be sure to append the '__c' after the
custom field name. Please reference your WSDL or the describe call for the appropriate names.</ErrorMessage>
  </Error>
</Salesforce>
```

Table-25: A sample error response XML due to an invalid field in SOQL query

Conclusion

This article presents a high level design pattern with step-by-step implementation guide including development of a generic messageflow for Salesforce cloud integration through IBM Integration Bus. Salesforce cloud integration through IBM Integration Bus is relatively expensive than IBM Websphere Cast Iron tool. But organizations with fewer integration requirements, already having IBM Integration Bus (or Websphere Message Broker) in their software stack can develop Salesforce cloud integration solution without buying Websphere Cast Iron or another integration tool.

Resources

- **IBM Integration Bus resources**

- IBM Integration Bus V9 information center
  A single Web portal to all IBM Integration Bus documentation, with conceptual, task, and reference information on installing, configuring, migrating to, and using IBM Integration Bus.
- IBM Integration Bus product page
  Product features, use cases, and resources.
- Technical article: What's new in IBM Integration Bus V9
  Replacing WebSphere Message Broker, IBM Integration Bus is IBM's strategic integration product for Java, Microsoft .NET, and heterogeneous integration scenarios. This article describes the highlights of IBM Integration Bus V9 for both existing WebSphere Message Broker and WebSphere ESB users, and for those new to integration.
- Announcement letter: IBM Integration Bus V9
  Official announcement information, including prerequisites, terms and conditions, and ordering information.
- Video: What's new in IBM Integration Bus
  A short YouTube video showing key IBM Integration Bus features.
- Information center topic: What's new in IBM Integration Bus for WebSphere ESB users
  Key differences between IBM Integration Bus and WebSphere ESB.
- Download IBM Integration Bus Developer Edition
  A lightweight edition that you can use for evaluation, development, unit test, and other scenarios.
- Follow IBM Integration Bus on Twitter
  Latest IBM Integration Bus news and announcements,
- IBM Integration Bus forum
  Forum on mqseries.net for user questions, answers, and tips.
- Track IBM Integration Bus user requirements
  Create, view, and track IBM Integration Bus user requirements.

- **WebSphere resources**
- developerWorks WebSphere
  Technical information and resources for developers who use WebSphere products. developerWorks WebSphere provides product downloads, how-to information, support resources, and a free technical library of more than 2000 technical articles, tutorials, best practices, IBM Redbooks, and online product manuals.
- developerWorks WebSphere application integration developer resources
  How-to articles, downloads, tutorials, education, product info, and other resources to help you build WebSphere application integration and business integration solutions.
- Most popular WebSphere trial downloads
  No-charge trial downloads for key WebSphere products.
- WebSphere forums
  Product-specific forums where you can get answers to your technical questions and share your expertise with other WebSphere users.
- WebSphere demos
  Download and watch these self-running demos, and learn how WebSphere products can provide business advantage for your company.
- WebSphere-related articles on developerWorks
  Over 3000 edited and categorized articles on WebSphere and related technologies by top practitioners and consultants inside and outside IBM. Search for what you need.
- developerWorks WebSphere weekly newsletter
  The developerWorks newsletter gives you the latest articles and information only on those topics that interest you. In addition to WebSphere, you can select from Java, Linux, Open source, Rational, SOA, Web services, and other topics. Subscribe now and design your custom mailing.
- WebSphere-related books from IBM Press
  Convenient online ordering through Barnes & Noble.
- WebSphere-related events
  Conferences, trade shows, Webcasts, and other events around the world of interest to WebSphere developers.

- **developerWorks resources**
  - Trial downloads for IBM software products
    No-charge trial downloads for selected IBM® DB2®, Lotus®, Rational®, Tivoli®, and WebSphere® products.
  - developerWorks business process management developer resources
    BPM how-to articles, downloads, tutorials, education, product info, and other resources to help you model, assemble, deploy, and manage business processes.
  - developerWorks blogs
    Join a conversation with developerWorks users and authors, and IBM editors and developers.
  - developerWorks tech briefings
    Free technical sessions by IBM experts to accelerate your learning curve and help you succeed in your most challenging software projects. Sessions range from one-hour virtual briefings to half-day and full-day live sessions in cities worldwide.
  - developerWorks podcasts
    Listen to interesting and offbeat interviews and discussions with software innovators.
  - developerWorks on Twitter
    Check out recent Twitter messages and URLs.
  - IBM Education Assistant
    A collection of multimedia educational modules that will help you better understand IBM software products and use them more effectively to meet your business requirements.