ORACLE®

PaaS Extending SaaS workshop
Advanced WebService Clients

Oracle Product Development

Safe Harbor Statement

The following is intended to outline our general product direction. It is intended for information purposes only, and may not be incorporated into any contract. It is not a commitment to deliver any material, code, or functionality, and should not be relied upon in making purchasing decisions. The development, release, and timing of any features or functionality described for Oracle's products remains at the sole discretion of Oracle.

Content Subject to Change

The information in this presentation is correct as of the recording date. However, Oracle Sales Cloud continues to evolve and software patches are applied frequently; therefore this information is subject to change. Check with your Oracle Representative for updates.

This content is not warranted to be error-free.

Topics

- Web Service proxies
- Outbound workflow
- Connector

Web service client libraries

- JAX-WS
  - The recommended library
- Apache CXF
  - Implements the JAX-WS APIs
- Apache Axis2
  - Web Services / SOAP / WSDL engine
- JDeveloper supports building of static SOAP Webservice proxies
  - .NET tooling has similar tooling

Web service libraries

JAX-WS

- JAX-WS allows for asynchronous communication
- JAX-WS is multi protocol compatible i.e. support of SOAP 1.1 and 1.2
- JAX-WS makes heavy of Java annotations as described by the JSR-181

Recap : Web service client types

Static Proxies

- Pro’s
  - Proxy compiled at design time
  - Collection of classes usually generated (e.g. JAX-B), helps with the development
  - Quick and easy for POCs and implementations where the server contract doesn’t change dramatically, or uses standard fields
- Cons
  - If a developer adds a field to the Oracle Sales Cloud then these fields are added to the WSDL. Unless the proxy is re-generated then these fields wont be seen by the code
  - JAXWS ignores the extra fields, AXIS/CXF have shown to throw errors

Recap : Web service client types

Dynamic Proxies

- Compile nothing at development time, all resolved at runtime
- Application retrieves WSDL, dynamically constructs calls
- More complex to develop
- Able to manage customizations and dynamicity in WSDL
- More robust for production use

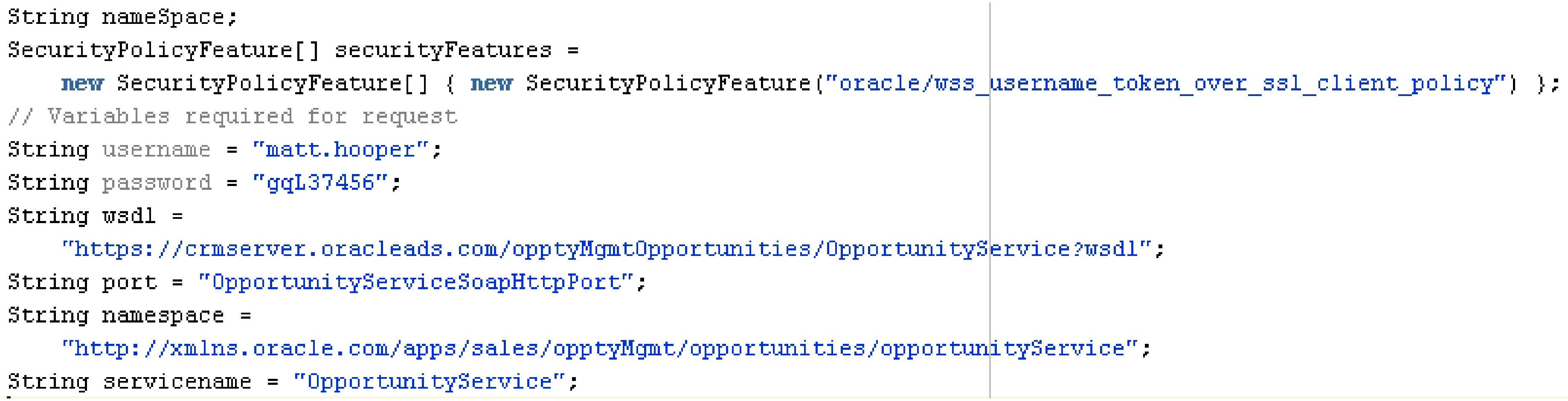
How to build a Dynamic proxy

Creating a Dispatch Instance

- Qualified name (QName) or endpoint reference of the target service endpoint.
- *Class of the type parameter T.*
- Usage mode
  - Message
  - Payload
- A list of Web service features to configure on the proxy
- JAXB context used to marshal or unmarshal messages or message payloads

Walk through : Building a dynamic proxy

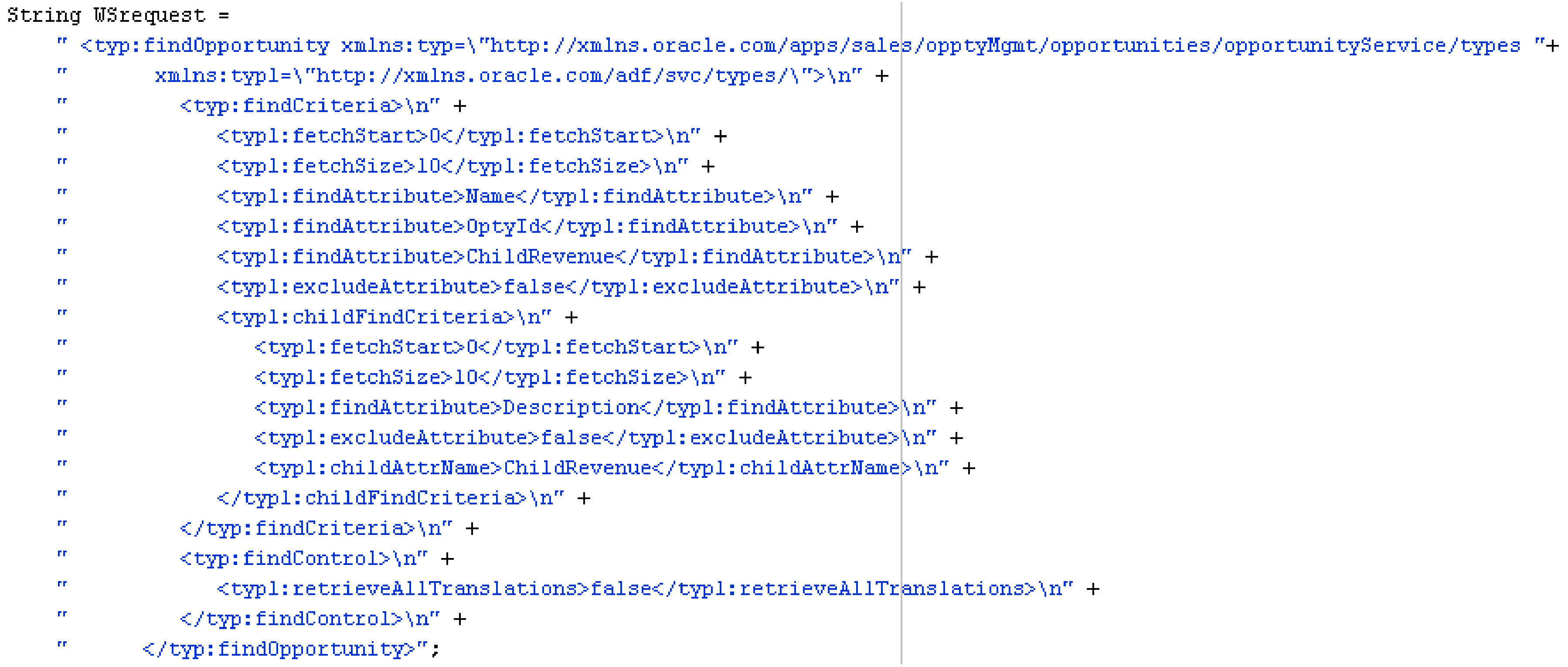
1. Define the variables



```
String nameSpace;
SecurityPolicyFeature[] securityFeatures =
    new SecurityPolicyFeature[] { new SecurityPolicyFeature("oracle/wss_username_token_over_ssl_client_policy") };
// Variables required for request
String username = "matt.hooper";
String password = "gqL37456";
String wsdl =
    "https://crmserver.oracleads.com/opptyMgmtOpportunities/OpportunityService?wsdl";
String port = "OpportunityServiceSoapHttpPort";
String namespace =
    "http://xmlns.oracle.com/apps/sales/opptyMgmt/opportunities/opportunityService";
String servicename = "OpportunityService";
```

Walk through : Building a dynamic proxy

2. Define the Payload



```
String WSrequest =
    " <typ:findOpportunity xmlns:typ=\"http://xmlns.oracle.com/apps/sales/opptyMgmt/opportunities/opportunityService/types "+
    "       xmlns:typ1=\"http://xmlns.oracle.com/adf/svc/types/\">\n" +
    "         <typ:findCriteria>\n" +
    "            <typ1:fetchStart>0</typ1:fetchStart>\n" +
    "            <typ1:fetchSize>10</typ1:fetchSize>\n" +
    "            <typ1:findAttribute>Name</typ1:findAttribute>\n" +
    "            <typ1:findAttribute>OptyId</typ1:findAttribute>\n" +
    "            <typ1:findAttribute>ChildRevenue</typ1:findAttribute>\n" +
    "            <typ1:excludeAttribute>false</typ1:excludeAttribute>\n" +
    "            <typ1:childFindCriteria>\n" +
    "               <typ1:fetchStart>0</typ1:fetchStart>\n" +
    "               <typ1:fetchSize>10</typ1:fetchSize>\n" +
    "               <typ1:findAttribute>Description</typ1:findAttribute>\n" +
    "               <typ1:excludeAttribute>false</typ1:excludeAttribute>\n" +
    "               <typ1:childAttrName>ChildRevenue</typ1:childAttrName>\n" +
    "            </typ1:childFindCriteria>\n" +
    "         </typ:findCriteria>\n" +
    "         <typ:findControl>\n" +
    "            <typ1:retrieveAllTranslations>false</typ1:retrieveAllTranslations>\n" +
    "         </typ:findControl>\n" +
    "      </typ:findOpportunity>";
```

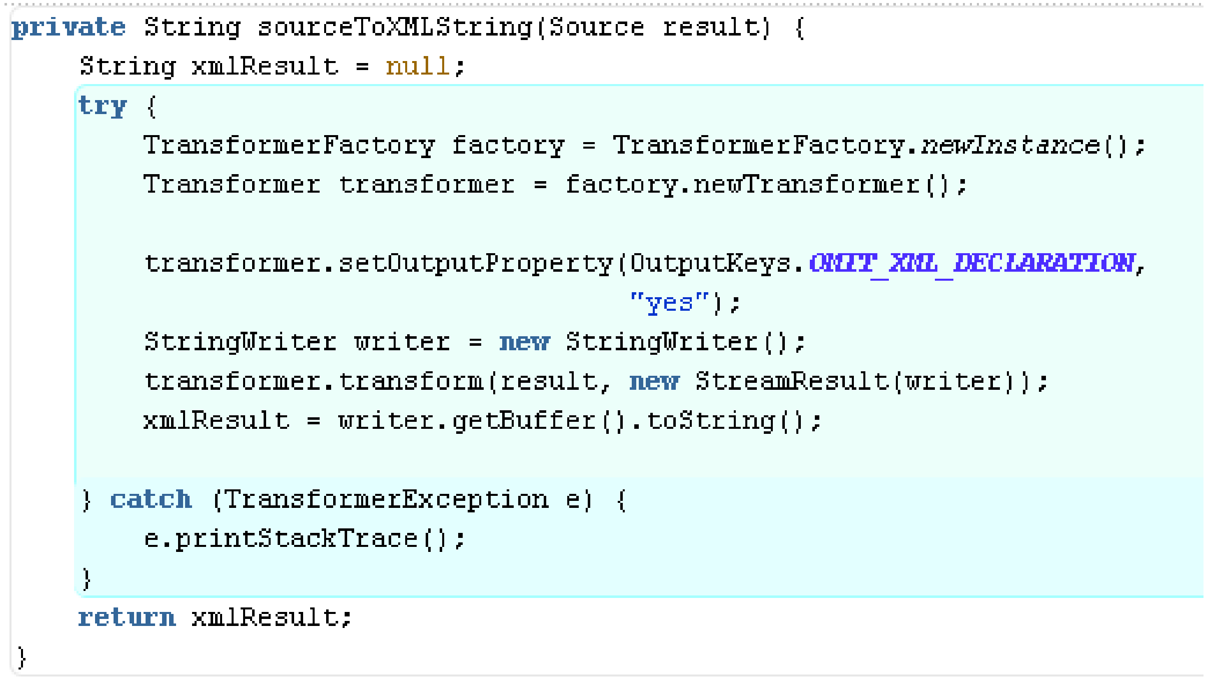
Walk through : Building a dynamic proxy

3. Put it together and execute the request

```
URL wsdlURL = new URL(wsdl);
Service service =
    Service.create(wsdlURL, new QName(namespace, servicename));
QName wsport = new QName(nameSpace, port);
Dispatch<Source> disp =
    service.createDispatch(wsport, Source.class, Service.Mode.PAYLOAD,
                           securityFeatures);
disp.getRequestContext().put(BindingProvider.USERNAME_PROPERTY,
                             "matt.hooper");
disp.getRequestContext().put(BindingProvider.PASSWORD_PROPERTY,
                             "password");

Source wsCallResult =
    disp.invoke(new StreamSource(new StringReader(WSrequest)));
String xmlResult = sourceToXMLString(wsCallResult);
System.out.println("Result from call " + xmlResult);
```

Walk through : Building a dynamic proxy

4. Helper function



```
private String sourceToXMLString(Source result) {
    String xmlResult = null;
    try {
        TransformerFactory factory = TransformerFactory.newInstance();
        Transformer transformer = factory.newTransformer();

        transformer.setOutputProperty(OutputKeys.OMIT_XML_DECLARATION,
                                      "yes");
        StringWriter writer = new StringWriter();
        transformer.transform(result, new StreamResult(writer));
        xmlResult = writer.getBuffer().toString();

    } catch (TransformerException e) {
        e.printStackTrace();
    }
    return xmlResult;
}
```

Challenges which determine the webservice proxy type

- Add new custom fields/custom child objects changes the payload
- Integrate with multiple Sales Cloud tenants- each having different customizations
- Publishing applications to Oracle Marketplace, you want to make sure your web service works on everyone's tenant instance

- Static Web Service proxies are not able to see new custom fields easily

Web service client types

JAX-WS Static Proxy with a handler

- Additional processing of the inbound and outbound message
- Provides methods to access and modify inbound and outbound messages
- Manage a set of properties
- Protocol handlers
  - Can access or change the protocol specific aspects of a message
- Logic handlers
  - Act only on the payload of the message

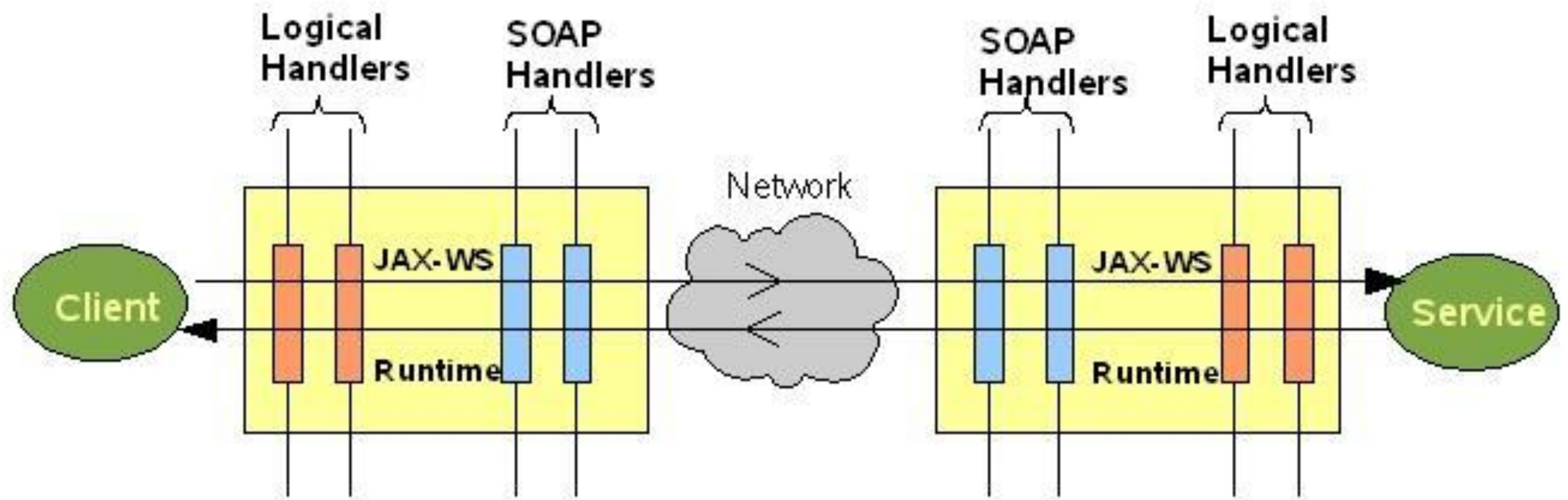
Web service client types

JAX-WS Static Proxy with a handler

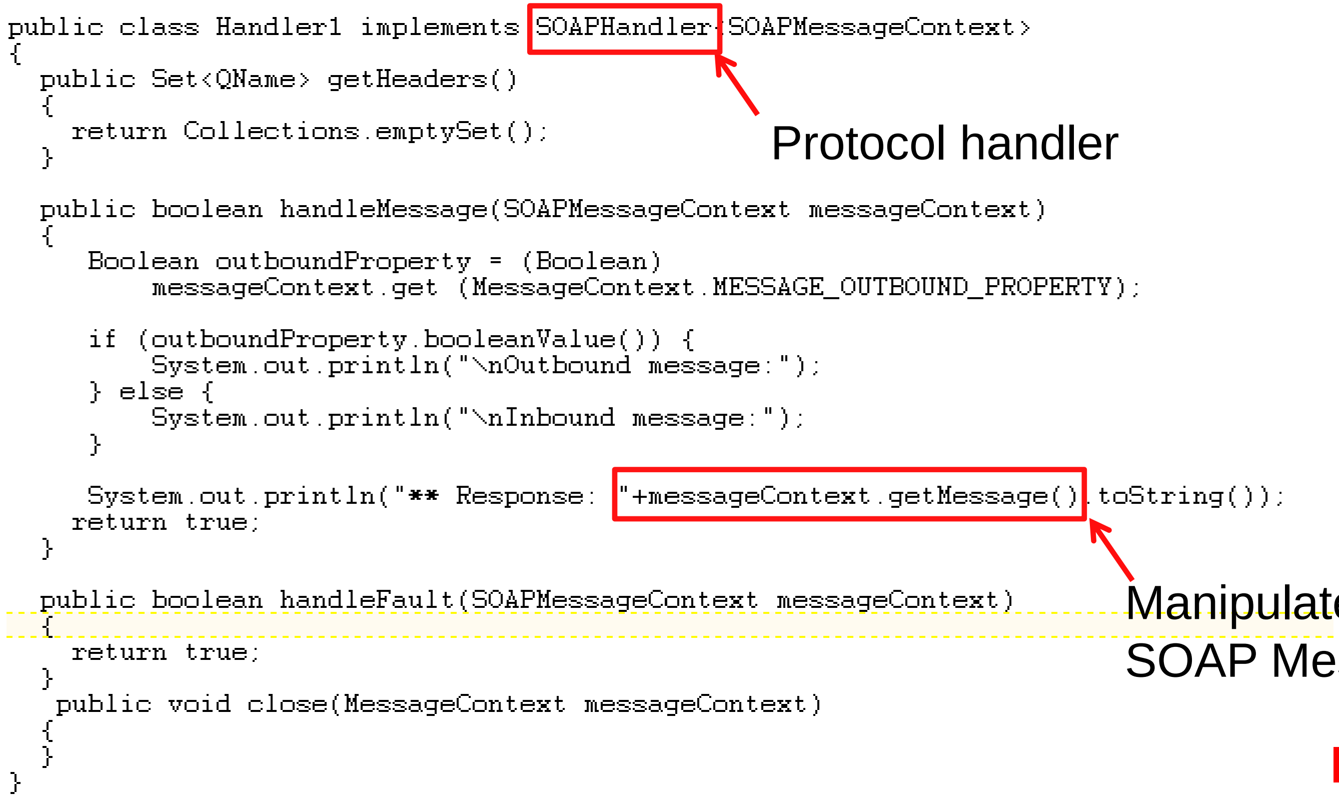
- No need for the full dynamic approach
- Custom fields handled from JAX-WS handler
- No need to regenerate static proxy for each tenant

Web service client types

JAX-WS Static Proxy with a handler



Example Code/How to build a Static proxy with Dynamic Hander



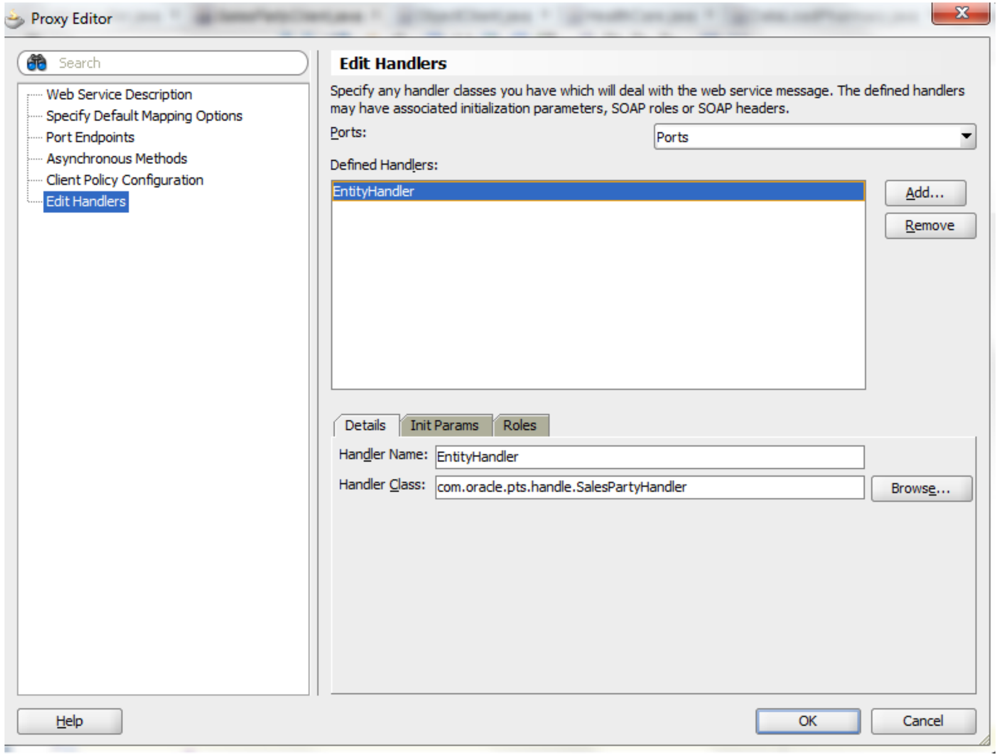
```
public class Handler1 implements SOAPHandler<SOAPMessageContext>
{
  public Set<QName> getHeaders()
  {
    return Collections.emptySet();
  }

  public boolean handleMessage(SOAPMessageContext messageContext)
  {
     Boolean outboundProperty = (Boolean)
         messageContext.get (MessageContext.MESSAGE_OUTBOUND_PROPERTY);

     if (outboundProperty.booleanValue()) {
         System.out.println("\nOutbound message:");
     } else {
         System.out.println("\nInbound message:");
     }

     System.out.println("** Response: "+messageContext.getMessage().toString());
    return true;
  }

  public boolean handleFault(SOAPMessageContext messageContext)
  {
    return true;
  }
   public void close(MessageContext messageContext)
  {
  }
}
```

Protocol handler

Manipulate the SOAP Message

Example Code/How to build a Static proxy with Dynamic Hander

- Configure handler through web service proxy
- Can add multiple handlers


Proxy Editor

Search

- Web Service Description
- Specify Default Mapping Options
- Port Endpoints
- Asynchronous Methods
- Client Policy Configuration
- Edit Handlers

Edit Handlers

Specify any handler classes you have which will deal with the web service message. The defined handlers may have associated initialization parameters, SOAP roles or SOAP headers.

Ports: Ports

Defined Handlers:

EntityHandler

Add... Remove

Details | Init Params | Roles

Handler Name: EntityHandler

Handler Class: com.oracle.pts.handle.SalesPartyHandler

Browse...

Help OK Cancel

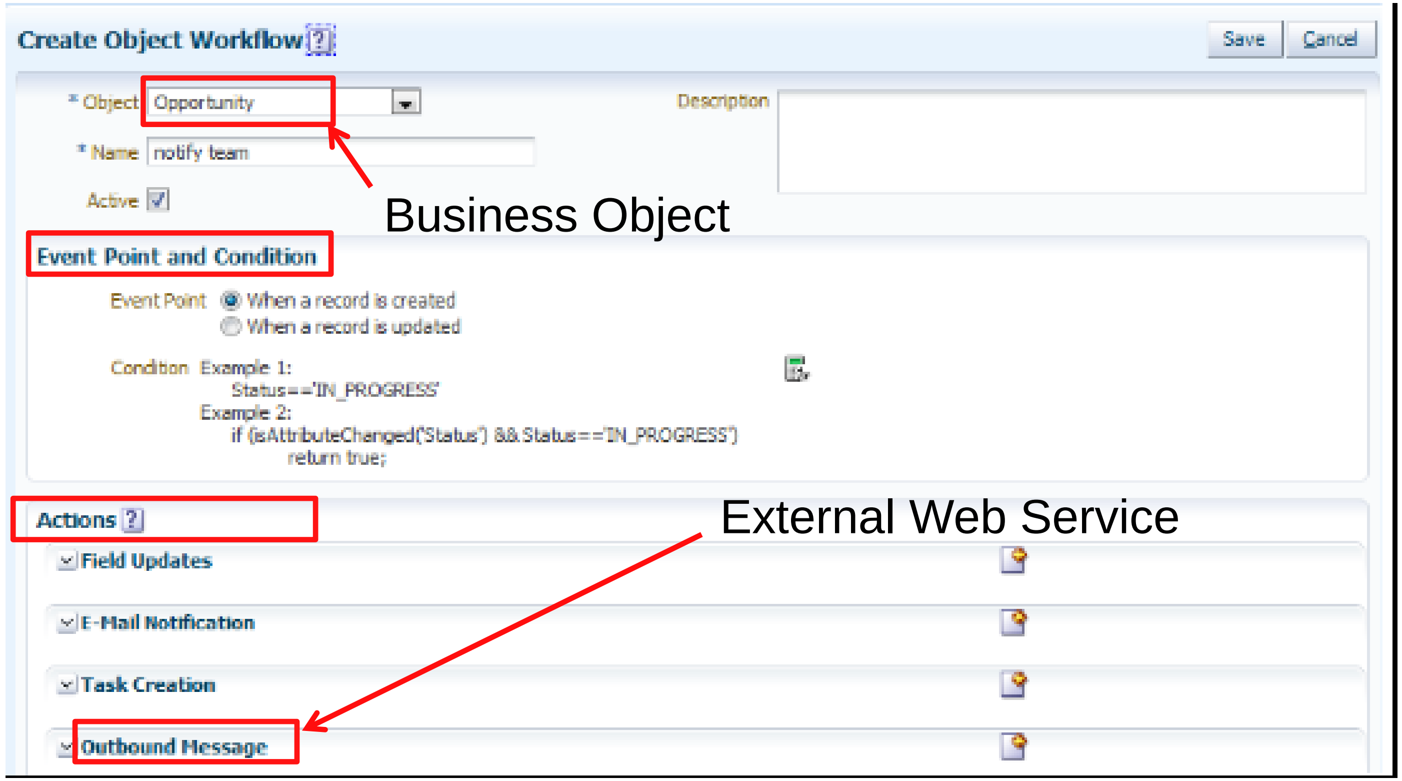
For more information

- http://docs.oracle.com/cd/E12839_01/web.1111/e13734/handlers.htm#i268373
- https://jax-ws.java.net/articles/handlers_introduction.html

Object Workflows within Sales Cloud

- Represent orchestrated business processes
- Business Object
  - Standard object delivered with the product or a custom object
- Event Point
  - When a record is created or updated
- Event Condition
  - Trigger for invoking object workflows
- Event Action
  - Field Updates, E-Mail Notification, Task Creation, Outbound Message

Object Workflow



* Object	Opportunity
* Name	notify team
Active	✓
Description	

Event Point and Condition

Event Point [✓ When a record is created] When a record is updated

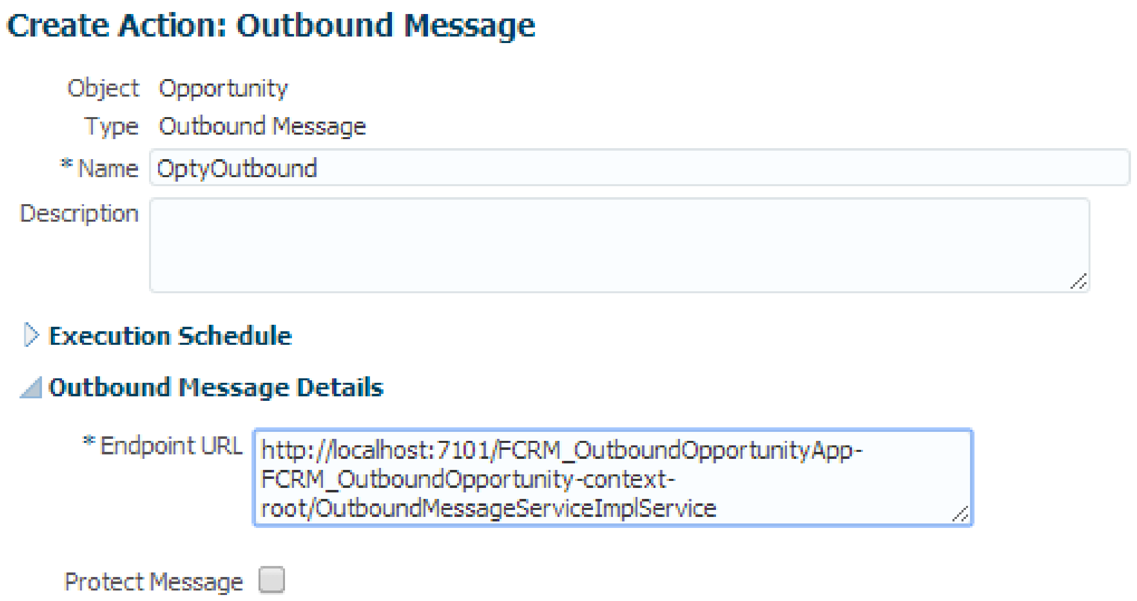
```
Condition  Example 1:
               Status=='IN_PROGRESS'
           Example 2:
               if (isAttributeChanged('Status') && Status=='IN_PROGRESS')
                    return true;
```

Actions

- Field Updates
- E-Mail Notification
- Task Creation
- Outbound Message

Outbound Message Service

- Add external web service endpoint
- Must conform to the service WSDL defined by Oracle Fusion



Create Action: Outbound Message

Object	Opportunity
Type	Outbound Message
* Name	OptyOutbound
Description	

Execution Schedule

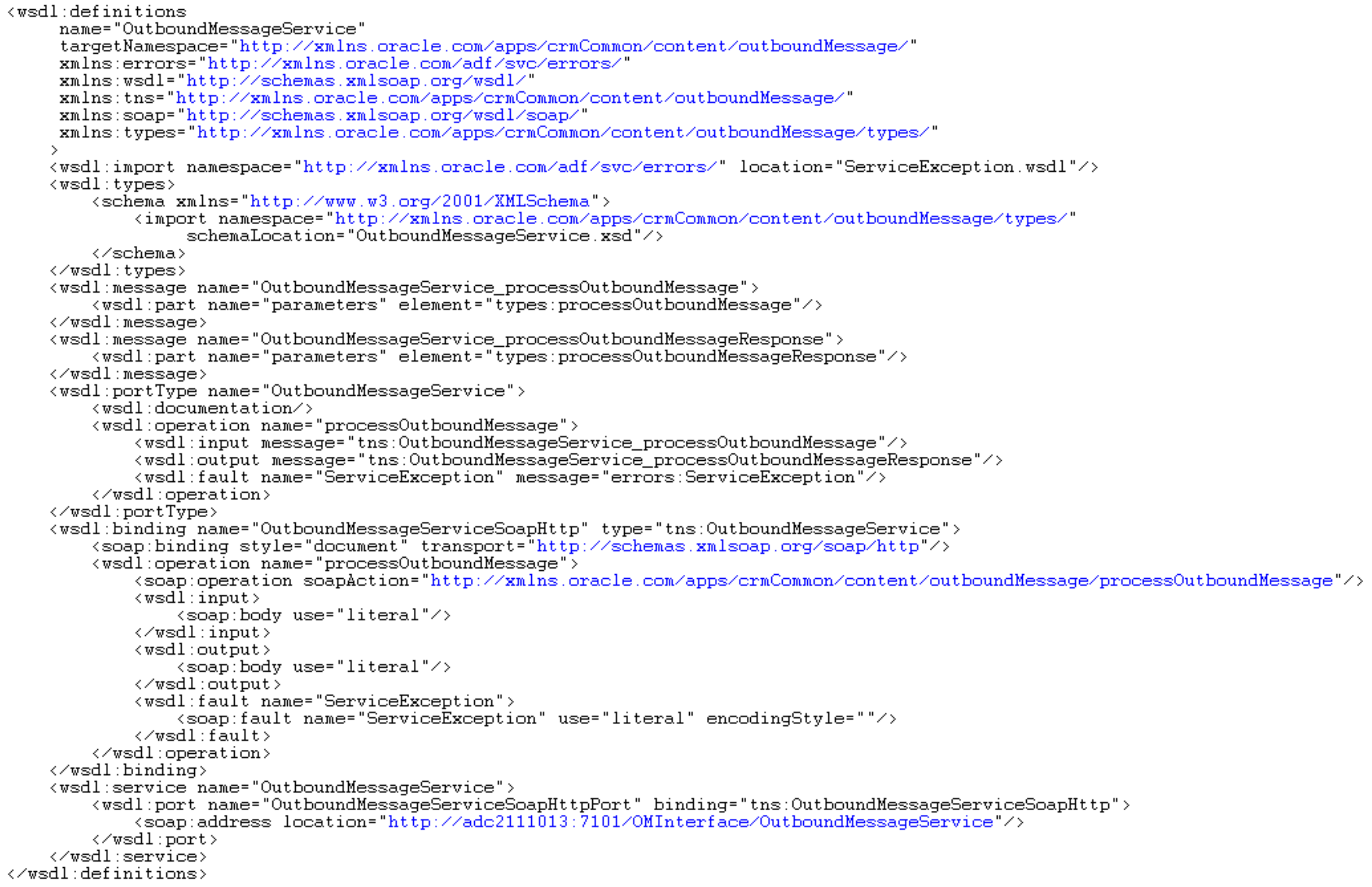
Outbound Message Details

* Endpoint URL	http://localhost:7101/FCRM_OutboundOpportunityApp-FCRM_OutboundOpportunity-context-root/OutboundMessageServiceImplService
Protect Message	☐

Creating Object-Specific Web Services

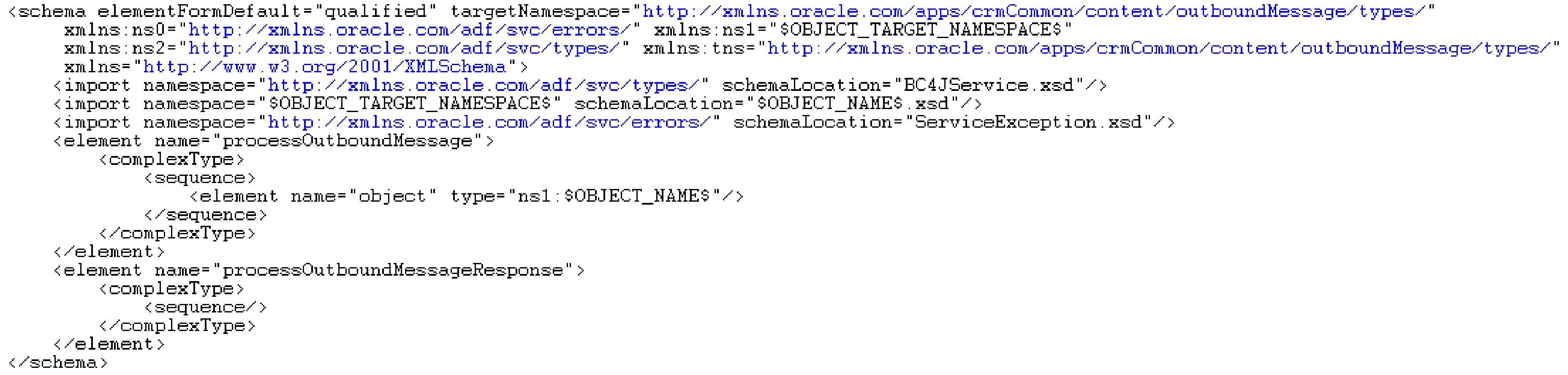
- For a standard object, search for **ADF Service** in OER by object name
- For custom objects, search for the generic Web service for all custom objects
- Extract the .xsd files from the live environment URL
- Replace the parameters in OutboundMessageService.xsd with the names for the object

WSDL File Example



```
<wsdl:definitions
     name="OutboundMessageService"
     targetNamespace="http://xmlns.oracle.com/apps/crmCommon/content/outboundMessage/"
     xmlns:errors="http://xmlns.oracle.com/adf/svc/errors/"
     xmlns:wsdl="http://schemas.xmlsoap.org/wsdl/"
     xmlns:tns="http://xmlns.oracle.com/apps/crmCommon/content/outboundMessage/"
     xmlns:soap="http://schemas.xmlsoap.org/wsdl/soap/"
     xmlns:types="http://xmlns.oracle.com/apps/crmCommon/content/outboundMessage/types/"
    >
    <wsdl:import namespace="http://xmlns.oracle.com/adf/svc/errors/" location="ServiceException.wsdl"/>
    <wsdl:types>
        <schema xmlns="http://www.w3.org/2001/XMLSchema">
            <import namespace="http://xmlns.oracle.com/apps/crmCommon/content/outboundMessage/types/"
                 schemaLocation="OutboundMessageService.xsd"/>
        </schema>
    </wsdl:types>
    <wsdl:message name="OutboundMessageService_processOutboundMessage">
        <wsdl:part name="parameters" element="types:processOutboundMessage"/>
    </wsdl:message>
    <wsdl:message name="OutboundMessageService_processOutboundMessageResponse">
        <wsdl:part name="parameters" element="types:processOutboundMessageResponse"/>
    </wsdl:message>
    <wsdl:portType name="OutboundMessageService">
        <wsdl:documentation/>
        <wsdl:operation name="processOutboundMessage">
            <wsdl:input message="tns:OutboundMessageService_processOutboundMessage"/>
            <wsdl:output message="tns:OutboundMessageService_processOutboundMessageResponse"/>
            <wsdl:fault name="ServiceException" message="errors:ServiceException"/>
        </wsdl:operation>
    </wsdl:portType>
    <wsdl:binding name="OutboundMessageServiceSoapHttp" type="tns:OutboundMessageService">
        <soap:binding style="document" transport="http://schemas.xmlsoap.org/soap/http"/>
        <wsdl:operation name="processOutboundMessage">
            <soap:operation soapAction="http://xmlns.oracle.com/apps/crmCommon/content/outboundMessage/processOutboundMessage"/>
            <wsdl:input>
                <soap:body use="literal"/>
            </wsdl:input>
            <wsdl:output>
                <soap:body use="literal"/>
            </wsdl:output>
            <wsdl:fault name="ServiceException">
                <soap:fault name="ServiceException" use="literal" encodingStyle=""/>
            </wsdl:fault>
        </wsdl:operation>
    </wsdl:binding>
    <wsdl:service name="OutboundMessageService">
        <wsdl:port name="OutboundMessageServiceSoapHttpPort" binding="tns:OutboundMessageServiceSoapHttp">
            <soap:address location="http://adc2111013:7101/OMInterface/OutboundMessageService"/>
        </wsdl:port>
    </wsdl:service>
</wsdl:definitions>
```

XSD File Example



```
<schema elementFormDefault="qualified" targetNamespace="http://xmlns.oracle.com/apps/crmCommon/content/outboundMessage/types/"
     xmlns:ns0="http://xmlns.oracle.com/adf/svc/errors/" xmlns:ns1="$OBJECT_TARGET_NAMESPACE$"
     xmlns:ns2="http://xmlns.oracle.com/adf/svc/types/" xmlns:tns="http://xmlns.oracle.com/apps/crmCommon/content/outboundMessage/types/"
     xmlns="http://www.w3.org/2001/XMLSchema">
    <import namespace="http://xmlns.oracle.com/adf/svc/types/" schemaLocation="BC4JService.xsd"/>
    <import namespace="$OBJECT_TARGET_NAMESPACE$" schemaLocation="$OBJECT_NAME$.xsd"/>
    <import namespace="http://xmlns.oracle.com/adf/svc/errors/" schemaLocation="ServiceException.xsd"/>
    <element name="processOutboundMessage">
        <complexType>
            <sequence>
                <element name="object" type="ns1:$OBJECT_NAME$"/>
            </sequence>
        </complexType>
    </element>
    <element name="processOutboundMessageResponse">
        <complexType>
            <sequence/>
        </complexType>
    </element>
</schema>
```

For more information

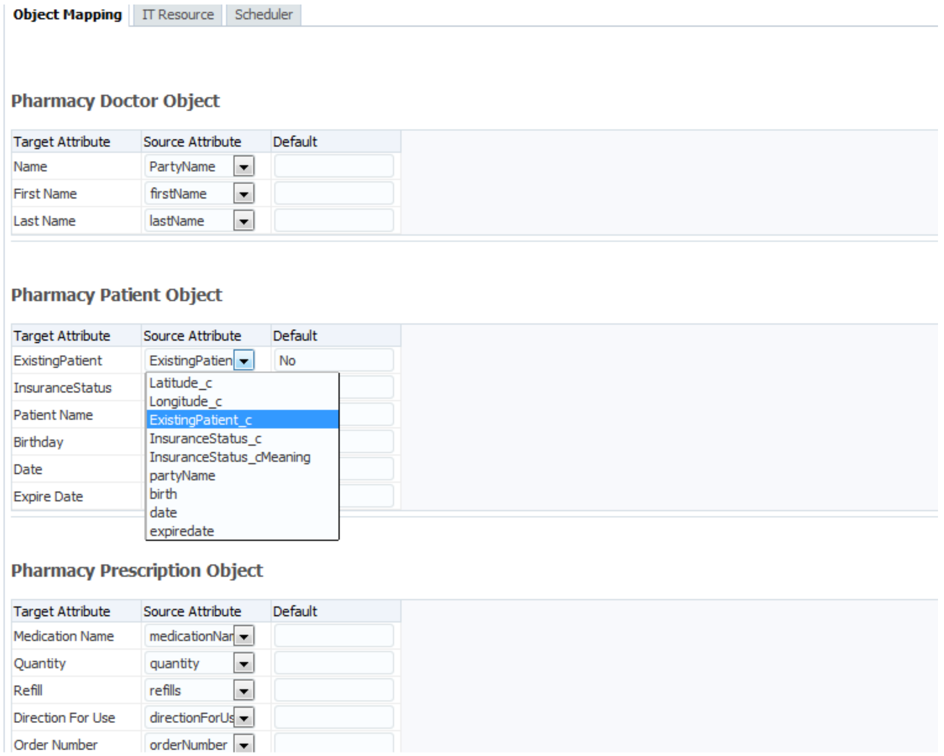
- http://docs.oracle.com/cloud/latest/salescs_gs/OACEX/F1071037AN24DB1.htm#F1078220AN24DE2

Sample Data Sync Connector

- Data sync between OSC/JCS and third party system
- Object mapping
- IT resource
- Scheduler
- Custom Field handling
- Latest version available on OTN Sample code http://www.oracle.com/technetwork/indexes/samplecode/cloud-samples-2203466.html

Sample Data Sync Connector

- Attribute mapping between target objects and source objects
- Custom fields dynamically retrieved from OSC
- Stored in DBCS



Pharmacy Doctor Object

Target Attribute	Source Attribute	Default
Name	PartyName	
First Name	firstName	
Last Name	lastName	

Pharmacy Patient Object

Target Attribute	Source Attribute	Default
ExistingPatient	ExistingPatien	No
InsuranceStatus		
Patient Name		
Birthday		
Date		
Expire Date		

Latitude_c
Longitude_c
ExistingPatient_c
InsuranceStatus_c
InsuranceStatus_cMeaning
partyName
birth
date
expiredate

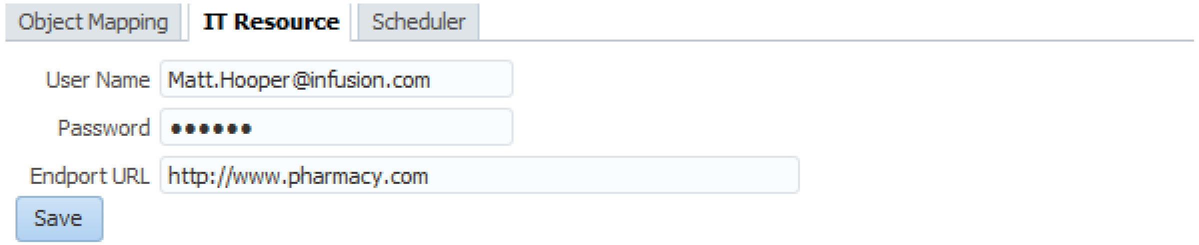
Pharmacy Prescription Object

Target Attribute	Source Attribute	Default
Medication Name	medicationNar	
Quantity	quantity	
Refill	refills	
Direction For Use	directionForUs	
Order Number	orderNumber	

Sample Data Sync Connector

IT Resource

- Connection info to the third party system
- Can be stored in a file or DBCS



User Name	Matt.Hooper@infusion.com
Password	●●●●●●
Endport URL	http://www.pharmacy.com

Sample Data Sync Connector

IT Resource

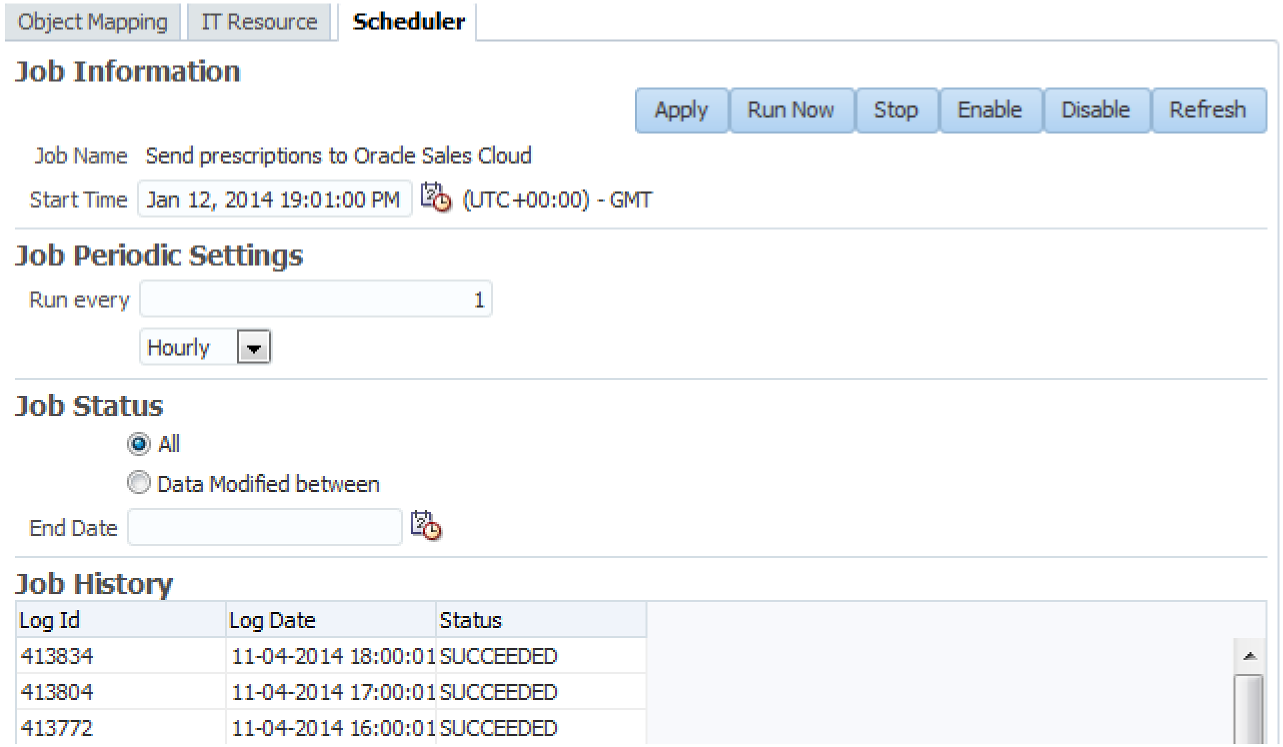
```
QName SERVICE_NAME =
    new QName("http://xmlns.oracle.com/apps/crmCommon/salesParties/salesPartiesService/",
              "SalesPartyService");
URL wsdlURL = null;
try {
    wsdlURL =
            new URL(FusionConfig.getInstance().getProperty("SALESPARTY_SERVICE") +
                    "?WSDL");
} catch (MalformedURLException e) {
    // TODO Auto-generated catch block
    e.printStackTrace();
}
salesPartyService_Service =
        new SalesPartyService_Service(wsdlURL, SERVICE_NAME);
SecurityPoliciesFeature securityFeatures =
    new SecurityPoliciesFeature(new String[] { securityPolicy });


salesPartyService =
        salesPartyService_Service.getSalesPartyServiceSoapHttpPort(securityFeatures);
```

End point retrieved from a configuration file or DBCS

Sample Data Sync Connector

Scheduler

- Data sync between OSC and the third party system
- Use DBCS scheduler
- Repeat interval and units
- Run/Stop/Enable/ Disable



Job Information

Job Name	Send prescriptions to Oracle Sales Cloud
Start Time	Jan 12, 2014 19:01:00 PM (UTC+00:00) - GMT

Job Periodic Settings

Run every	1
	Hourly

Job Status

[● All] ○ Data Modified between

End Date

Job History

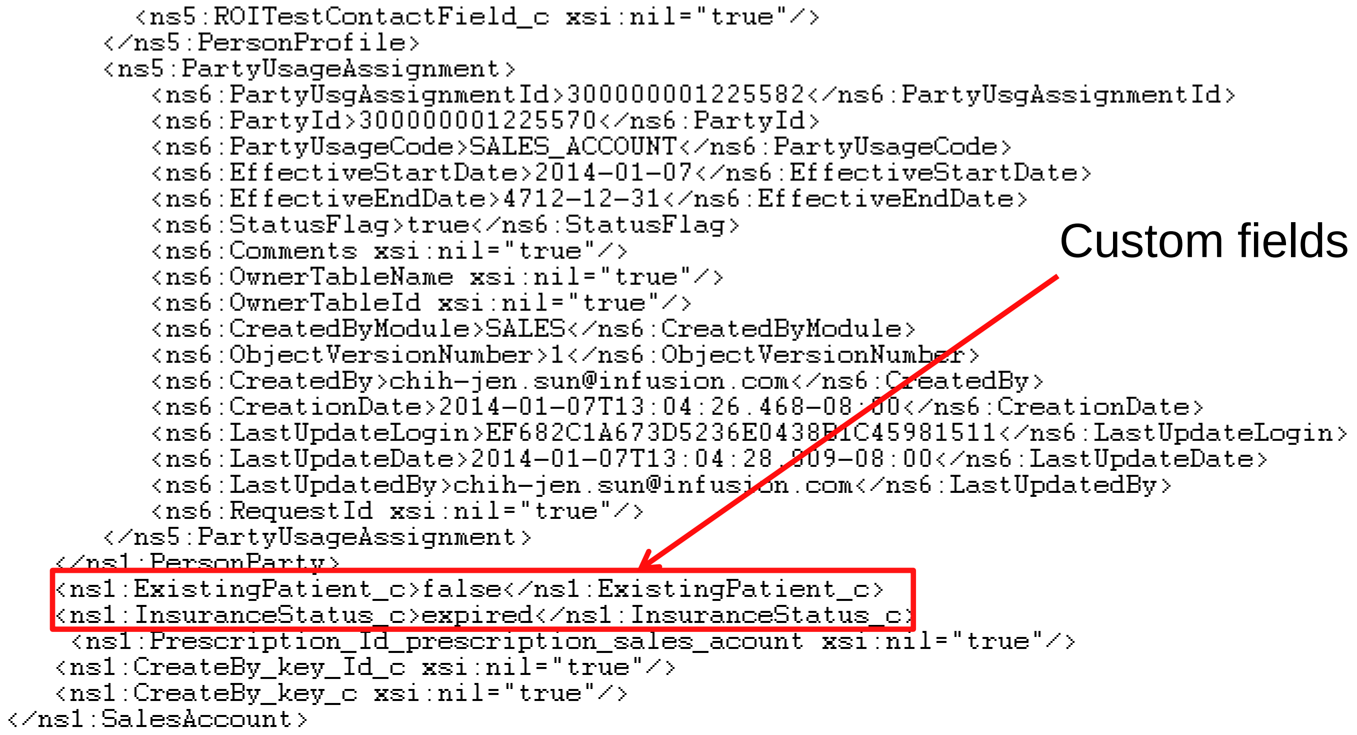
Log Id	Log Date	Status
413834	11-04-2014 18:00:01	SUCCEEDED
413804	11-04-2014 17:00:01	SUCCEEDED
413772	11-04-2014 16:00:01	SUCCEEDED

Sample Sync Connector

Custom Field handling

- Use JAX-WS handler + static proxy
- No need to regenerate static proxy
- Custom field definition are configured OSC at run time.
- Handler parse payload and set/get custom field value

Example Payload



```
        <ns5:ROITestContactField_c xsi:nil="true"/>
      </ns5:PersonProfile>
      <ns5:PartyUsageAssignment>
         <ns6:PartyUsgAssignmentId>300000001225582</ns6:PartyUsgAssignmentId>
         <ns6:PartyId>300000001225570</ns6:PartyId>
         <ns6:PartyUsageCode>SALES_ACCOUNT</ns6:PartyUsageCode>
         <ns6:EffectiveStartDate>2014-01-07</ns6:EffectiveStartDate>
         <ns6:EffectiveEndDate>4712-12-31</ns6:EffectiveEndDate>
         <ns6:StatusFlag>true</ns6:StatusFlag>
         <ns6:Comments xsi:nil="true"/>
         <ns6:OwnerTableName xsi:nil="true"/>
         <ns6:OwnerTableId xsi:nil="true"/>
         <ns6:CreatedByModule>SALES</ns6:CreatedByModule>
         <ns6:ObjectVersionNumber>1</ns6:ObjectVersionNumber>
         <ns6:CreatedBy>chih-jen.sun@infusion.com</ns6:CreatedBy>
         <ns6:CreationDate>2014-01-07T13:04:26.468-08:00</ns6:CreationDate>
         <ns6:LastUpdateLogin>EF682C1A673D5236E0438B1C45981511</ns6:LastUpdateLogin>
         <ns6:LastUpdateDate>2014-01-07T13:04:28.909-08:00</ns6:LastUpdateDate>
         <ns6:LastUpdatedBy>chih-jen.sun@infusion.com</ns6:LastUpdatedBy>
         <ns6:RequestId xsi:nil="true"/>
      </ns5:PartyUsageAssignment>
   </ns1:PersonParty>
   <ns1:ExistingPatient_c>false</ns1:ExistingPatient_c>
   <ns1:InsuranceStatus_c>expired</ns1:InsuranceStatus_c>
    <ns1:Prescription_Id_prescription_sales_acount xsi:nil="true"/>
   <ns1:CreateBy_key_Id_c xsi:nil="true"/>
   <ns1:CreateBy_key_c xsi:nil="true"/>
</ns1:SalesAccount>
```

Custom fields end with _c

Sample Data Sync Connector

Custom Field handling

```
  if (cNodeName.equals("result")) {
        CustomFieldHolder customFieldHolder = new CustomFieldHolder();
        List<DataSet> dataSetList = customFieldHolder.getDataSetList();
        DataSet dataSet = new DataSet();
        dataSet.setName(customFieldHolder.getObjectName());
        String keyName = CustomFieldHolder.getKeyName(dataSet.getName());
        NodeList ccNodeList = cNode.getChildNodes();
        for (int k = 0; k < ccNodeList.getLength(); k++) {
            Node ccNode = ccNodeList.item(k);
            String ccNodeName = ccNode.getNodeName();
            ccNodeName =
                    ccNodeName.substring(ccNode.getNodeName().indexOf(":") +
                                         1, ccNodeName.length());
                MetaInfo cMetaInfo =   metaInfo.getChildMetaInfo(ccNodeName);
                if (cMetaInfo!=null ) {// child
                      DataSet cDataSet = dataSet.getChildDataSetByName(ccNodeName);
                      processData(cDataSet,ccNode,cMetaInfo);
                }
                else{ // attribute
                if(keyName.equals(ccNodeName)){
                    dataSet.setId(ccNode.getTextContent());
                }

                    if (ccNodeName.contains("_c")) {
                        AttributeEntry attr = new AttributeEntry();
                        attr.setName(ccNodeName);
                        attr.setValue(ccNode.getTextContent());
                        dataSet.getAttributeList().add(attr);
                    }
                }
             }
        }
        dataSetList.add(dataSet);
        SOAPElement soapElement = (SOAPElement)cNode;
    }
```

It is custom field. Extract it and add it to dataSet

Demo
Sample Data Sync Connector

ORACLE®