ORACLE

Oracle Fusion Applications Development and Extensibility Handbook

ORACLE FUSION APPLICATIONS

Create and Customize High-Performance Business Applications

Vladimir Ajvaz
Anil Passi
Dhaval Mehta

Oracle Press™

Excerpted from *Oracle Fusion Applications Development and Extensibility Handbook* by **Vladimir Ajvaz**, **Anil Passi, and Dhaval Mehta**

Oracle Fusion Applications Development and Extensibility Handbook contains best practices, real-world case studies, and technical deep dives. Discover how to manage design- and run-time customizations, extend existing UIs and build new ones, secure your applications, and integrate with other systems. This Oracle Press guide offers complete coverage of the latest cloud and SOA-based features.

- Explore Oracle Fusion Applications components and architecture
- Plan, develop, debug, and deploy customizations
- Extend out-of-the-box functionality with Oracle JDeveloper
- Modify web applications using Oracle Composer
- Incorporate Oracle SOA Suite 11g composites
- Validate code through sandboxes and test environments
- Secure data using authorization, authentication, and encryption
- Design and distribute personalized BI reports
- Automate jobs with Oracle Enterprise Scheduler
- Change appearance and branding of your applications with the Oracle ADF Skin Editor
- Extend and customize CRM with Application Composer

Vladimir Ajvaz is an SOA architect at the prestigious Imperial College in London where he uses Oracle Fusion technologies to integrate various systems with Oracle E-Business Suite. Previously, Vladimir was a Senior Consultant in the Development Technology Practice of Oracle Consulting Services (Oracle Corp).

Anil Passi is an Oracle ACE with more than 10 years of technical consulting experience. He has worked on many large-scale Oracle Applications implementations across Europe as an independent consultant. Anil is a well-known speaker on Oracle Applications, and gives best practice seminars across Europe on development and extensions of Oracle Applications. He also runs http://apps2fusion.com knowledge portal to help Oracle E-Business suite developers transition to Oracle Fusion Applications.

Dhaval Mehta is a Group Development Manager for Fusion Applications in Oracle Corporation. Previously Dhaval worked on Oracle E-Business Suite applications development within Oracle Corporation. Dhaval has worked on building Oracle Sales Cloud applications from their inception to current releases.

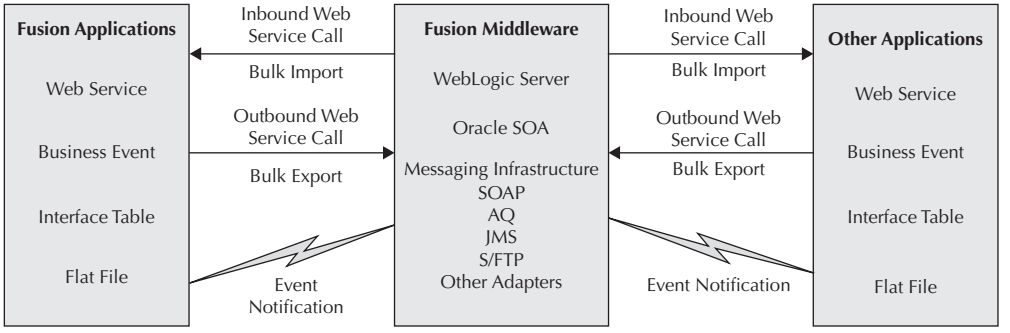
OraclePressBooks.com
Amazon.com

CHAPTER 15

Integration with Fusion Applications

Fusion Applications are completely built from scratch and you have to implement them from the ground up. You will find that the features and functions in Fusion Applications may not be exactly the same as what you are using in other legacy applications such as Oracle E-Business Suite, PeopleSoft, Siebel, or any other non-Oracle application suite. Most of the time, you cannot simply shut down your existing applications and immediately move to use Fusion Applications in a production instance. Sometimes you may not find all necessary functions and products in Fusion Applications and you may have to continue to use your legacy application and Fusion Applications at the same time. You have to migrate your data from existing systems to a new Fusion Applications instance. All of this means that there is a need to integrate Fusion Applications with other non-Fusion applications or other systems. In this chapter, we will discuss some of the integration patterns and best practices of how you can integrate with Fusion Applications.

You can integrate with Fusion Applications to exchange data both inbound and outbound. You can also integrate using real-time or batch mode to exchange information. Different products have different capabilities, but the general strategy for integration remains the same. For inbound, you will use Web services or bulk import tools, and for outbound, you will use business events or bulk export tools. You can use your choice of middleware for integrations because Fusion Applications are built on standards-based Fusion Middleware. The following diagram illustrates the general interaction pattern for Fusion Applications along with Fusion Middleware.



What Is Oracle Enterprise Repository (OER)?

Fusion Applications document all the integration artifacts or assets in Oracle Enterprise Repository (OER). You can use OER as a single source of integration information. It provides visibility into lifecycle and support details about a given asset. Each asset is marked with the level of compatibility supported by Oracle. You can discover all assets in OER, including the ones that are not supported.

Fusion Applications are designed with Service Oriented Architecture in mind. There are many services in Fusion Applications, but not all of them are suited for integration purposes. You should only use assets that are marked as Compatibility = "Supported – Backward Compatibility Assured" for integrations so that upgrades or patches do not break your code. The services in OER are also marked with the keyword EXTERNAL to indicate that the service end point is visible to external clients. Table 15-1 explains the intended usage of assets with a given compatibility.

Compatibility	Keyword	Usage
Supported	EXTERNAL	Services are available to external clients. Services can be used by customers and partners to extend and integrate with Fusion Applications.
Supported	INTERNAL or not specified	Services cannot be accessed by external clients but can only be accessed by custom composites that are deployed in Fusion Applications SOA domains.
Not Supported	EXTERNAL	These series are accessible to external clients but not designed to be used for integration purposes by customers or partners. Such services are made external for specific out-of-the-box integration Fusion Applications may have as part of standard functionality. Oracle can change the interfaces at any time without notice to customers.
Not Supported	INTERNAL or not specified	Services cannot be accessed by external clients and should not be used even by custom composites that are deployed in Fusion Applications SOA domains. Oracle can change the interfaces at any time without notice to customers.

TABLE 15-1. *Customization Layers*

What Are the Different Types of Assets in OER?

OER documents all Fusion Applications assets. In this section, we will describe some of the important types of these assets that you will use for integration purposes.

Web Services

Fusion Applications use standards-based Web services to allow inbound integrations. Any development environment or tool that is compliant with Web service standards can be used to invoke Fusion Applications Web services. Fusion Applications expose two types of Web services depending on their underlying implementation. Both the services are exposed in OER—ADF Services and SOA Composite services.

ADF Services

ADF Services are built on Fusion Applications business components. These services expose standard Create, Read, Update, and Delete (CRUD) type of operations to manipulate the data for a given object. ADF Services also expose special operations to do a specific business function such as convert a lead to an opportunity or promote an employee. These services allow you to access the data using Service Data Object (SDO) where the entire object structure is exposed to the service interface using a schema (XSD). The standard SDO-based ADF service for a given object will have create, update, delete, get, find, and merge operations that you can use to query and manipulate the data for a given object. The merge operation is a combination of insert and update in which the data is created if the row is not found for the given identifier. The find operation is a very powerful tool to return the data based on your filter conditions as well as attributes that you like to see in the output of the service invocation. All the operations and input/output schema for a given service are documented in OER and can be seen from the Web service definition file (WSDL) as well.

SOA Composite Services

A composite service typically represents a complete business flow such as order fulfillment. This service might be triggered by certain actions or functions within Fusion Applications or can be invoked explicitly by a program or client. Composite services may be non-object-based and represent an end-to-end back-end process that spans multiple objects and orchestrates other ADF Services, human task workflow, or business rules. The composite services are also described by WSDL and XSD and can be invoked by any standard Web service client, similar to the way you invoke

ADF Services. Fusion Applications use composite services when there is an out-of-the-box integration between different functions and the integration needs to be loosely coupled between those applications. Typically these composites are triggered by raising business events in one application and use ADF Services to communicate between these applications.

Business Events

Fusion Applications business objects publish business events to the Event Delivery Network (EDN). Business events are a way to notify any subscriber about any change to a given business object. Every object in Fusion Application does not raise business events; you can check product-specific documentation to find out if there are events that are not documented in OER. The business events for a given object could be raised on create, update, or delete of a given instance of the business object. There are business process–specific events such as bulk import as well that are available in Fusion Applications.

Scheduled Processes

Fusion Applications use Enterprise Scheduler Services (ESS) for background processing. You can find these ESS programs in OER and understand details such as their use, parameters required to run the process, and so on. Some of the ESS programs are exposed as a task in FSM so that the administrator does not need to understand the details behind it. Some are invoked internally from the UI or from SOA composites. Please read Chapter 13 for more details on ESS.

Tables and Views

All the physical tables and views for Fusion Applications are available in OER. These tables are used to store the transaction, setup, and configuration data. OER exposes all important information such as the table description, columns, column description, index, constraints on the table, and so on. Most of the important Fusion Applications objects support bulk import, which is useful for initial data load or migration or periodic bulk updates. Typically, data loads are done through interface tables where you populate the data in raw format and then the Fusion Applications import program will move the data into transaction tables. In the process, it will apply the necessary data conversions and validation rules to make sure there are no data integrity issues. You can load the interface tables in a variety of ways such as Oracle Data Integrator (ODI), a database adapter in Oracle SOA Suite, or SQL commands or loaders.

Data Model Diagrams

In addition to physical table information, OER also exposes data model diagrams. You can find the diagram for a given logical area. The diagram shows relationships between physical tables and helps you understand internal implementation. OER exposes both a logical data model diagram and a physical relational data model diagram.

How to Discover Integration Assets in OER

To view OER on your deployment, navigate to http:/host:port/oer where you have installed OER. This gives you information based on the Fusion Applications release you have provisioned and gives you a concrete URL for WSDL and XSD. You can also go to the public OER hosted by Oracle at https://fusionappsoer.oracle.com if you just want to discover assets. This publicly hosted OER is for the latest Fusion Applications release that is available for download from Oracle e-Delivery. You can log in to OER, either as a guest or with an authenticated account as shown in Figure 15-1.

Once you log in to OER, you can find what is new in the current release, how to find documentation for various products, and how to search assets in OER as shown in Figure 15-2.

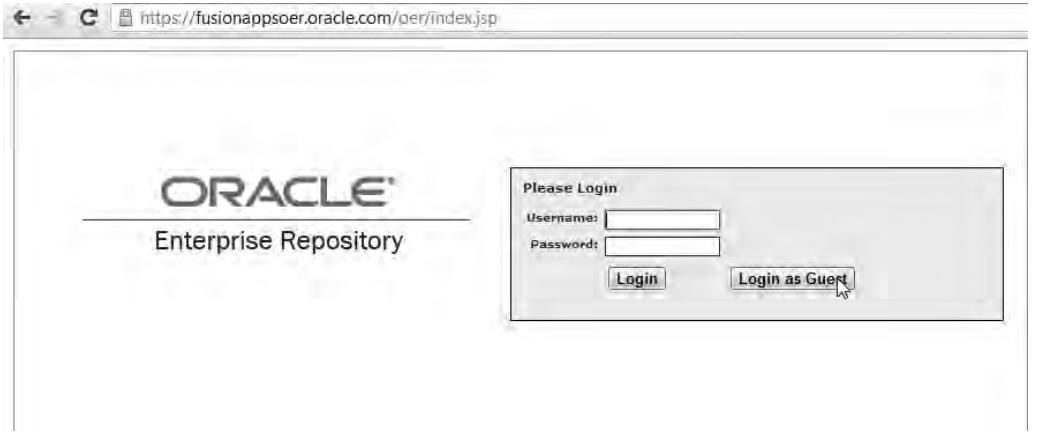


FIGURE 15-1. *OER login screen*

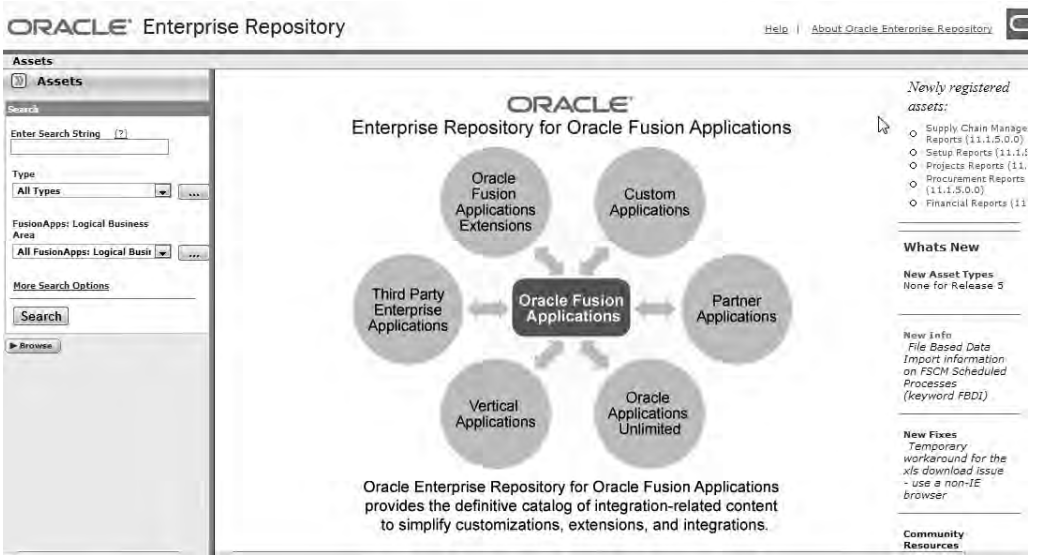
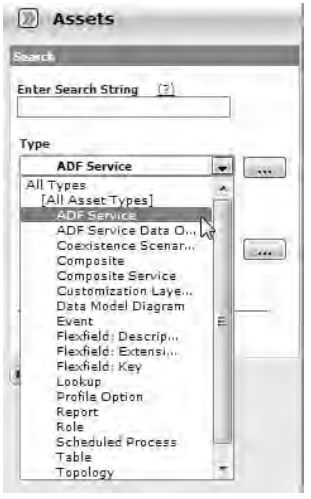


FIGURE 15-2. *OER home page*

We will search for a Web service to manage location data in the customer data management module using OER in this section.

1. From the Assets search window on the left-hand side, select Type as ADF Service as shown in the following illustration.



2. Select Customer Data Management as Logical Business Area.

3. Provide the search string as **location** and click the Search button. This will return the matching services for the given criteria as shown here.

4. You can examine the details about the selected service on the Overview tab. It provides a description of the service, its lifecycle, and compatibility details as shown in Figure 15-3.

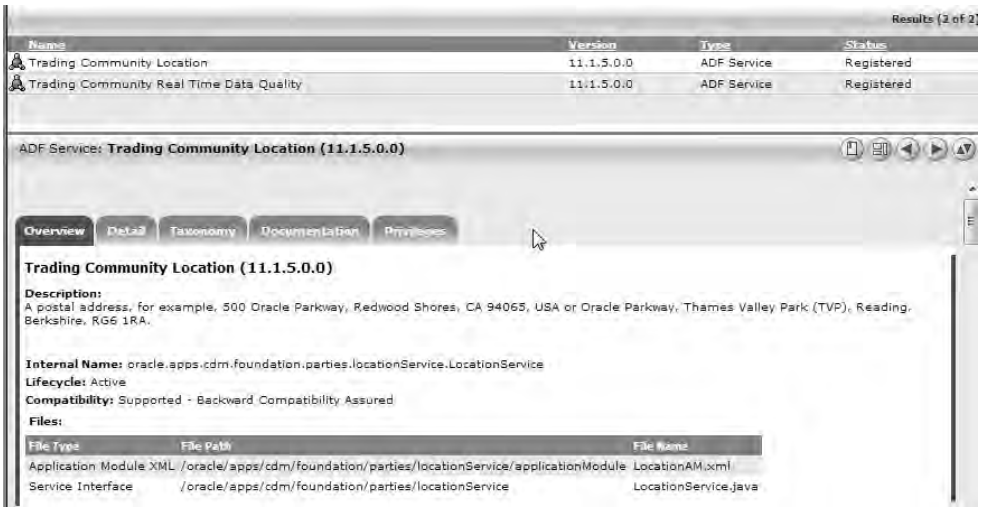


FIGURE 15-3. *Location service details*

5. You can examine the operations, their description, and input and output parameters for this service on the Detail tab, as shown in Figure 15-4.

6. The Detail tab also shows you the WSDL location for this service as shown here.

Service Path: http://<HostName:PortNumber>/foundationParties/LocationService?WSDL
Abstract WSDL URL: rep://FUSIONAPPS_HOSTEDREP/oracle/apps/cdm/foundation/parties/locationService/LocationService.wsdl

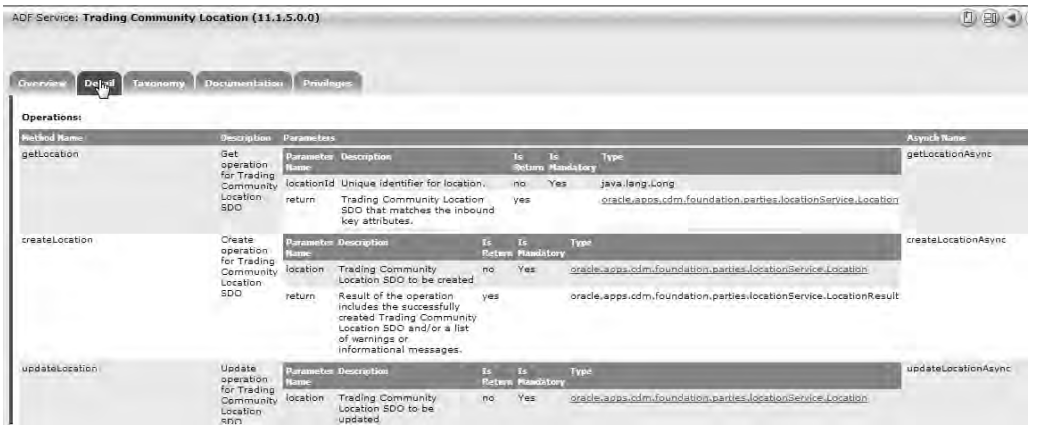


FIGURE 15-4. *Location service operations*

FIGURE 15-5. *Location service documentation*

7. The Documentation tab allows you to check the XSD for this Web service and its operations. It also provides you a link to the cookbook on how the service can be invoked and complete details about the service from OER, as shown in Figure 15-5.
8. The Privileges tab gives details about what privileges and roles the user will need to access the service and its operations, as shown in Figure 15-6.
9. Similarly, you can search for other types of assets and examine all the details about the artifact and how you can use that for your integrations using OER.

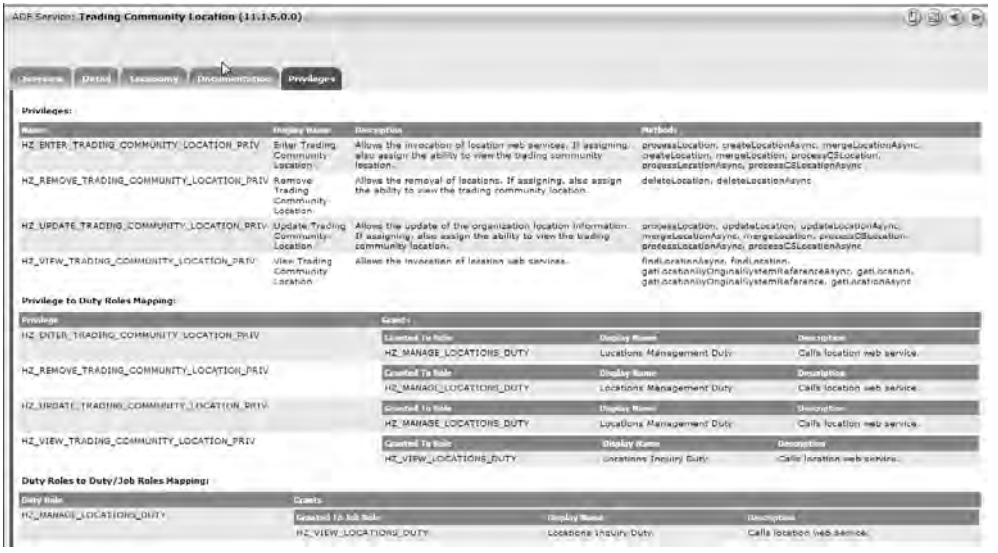


FIGURE 15-6. *Location service security*

Outbound Integration Patterns with Fusion Applications

You may need to call out to other applications from Fusion Applications when your business requirements span multiple applications. Outbound calls may be needed for a totally new business process or as a customized or extended process to Fusion Applications. Different precuts in Fusion Applications provide different capabilities to do outbound integrations. We will briefly discuss these options in the following section.

Object Workflow

CRM Application Composer provides you a feature called Object Workflow as discussed in Chapter 6. This feature allows you to define your own events and triggers on several out-of-the-box objects in CRM. In response to the event you define, you can choose an outbound service call as an action. You need to first use OER or product-specific documentation to find out the Web service interface for a given object for outbound call and then build your own WS that takes the same XSD as input. Once you have your own service that take a CRM object as input, you register that service with a CRM object workflow outbound call in response to the event for that object. At run time, when the condition specified in Object Workflow is satisfied, it will call out the Web service you registered. Your Web service will need to do whatever action you may need per your requirement to take the object details and communicate to other applications. The CRM Object Workflow is the only mechanism that lets you do real-time outbound integration with Fusion Applications in SaaS deployment mode. This option works for On-Premise customers as well.

Business Events

Fusion Applications provide many business events on several objects. These events are raised on Fusion Applications SOA domain EDN for a given product family such as CRM or HCM. Since these events are raised within the Fusion Application domain, SaaS customers cannot subscribe to these events. You can consume these events if you have access to middleware domains in your On-Premise deployment. You can use OER to find out what business events are available for a given area. Some events may not be documented, so you may have to read product-specific documentation for that. Once you find out the event details, you can create an SOA Composite With Mediator that will subscribe to the event using the Event Definition Language (EDL) file. The event payload should give you enough information to then get more details about the object and do necessary processing with that data in your composite. If there are no existing events that meet your needs, you can use JDeveloper as discussed in Chapter 7 to add your own custom events and subscribe to them in your Mediator. Business events are the standard and recommended approach to do outbound integration with Fusion Applications.

Bulk Export

There are certain Fusion Applications products that provide Bulk Export interfaces, which allow you to extract the data out of Fusion Applications. Fusion CRM applications have a "Schedule Export Processes" FSM task that allows you to select an object you want to export, the filter criteria, the attributes to export, and the scheduling frequency. You can use this task to get the data out of CRM applications either on SaaS or On-Premise deployment. Figure 15-7 shows an example of exporting location data with a few selected attributes.

Similarly, HCM applications have a Bulk Extract interface that allows you to specify data blocks to be extracted for given logical entity. It allows you to specify extract parameters, delivery options, delivery schedule, and other mappings that you can use for reporting purposes. You can access the HCM extract UI using the FSM task "Define Extracts." Figure 15-8 shows an example payroll extract. Please read HCM product documentation for more details on all the configuration and extract options and capabilities.

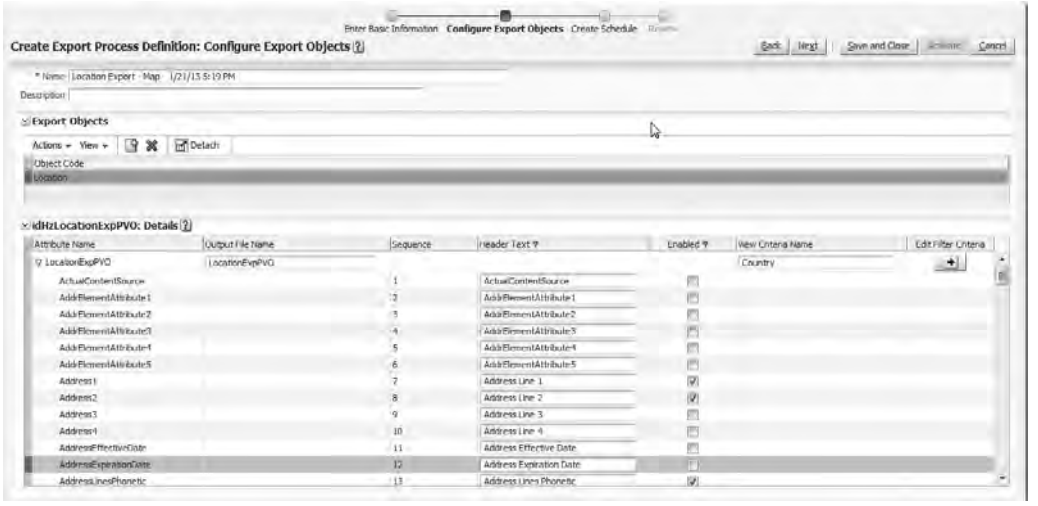


FIGURE 15-7. *Bulk Export configuration*

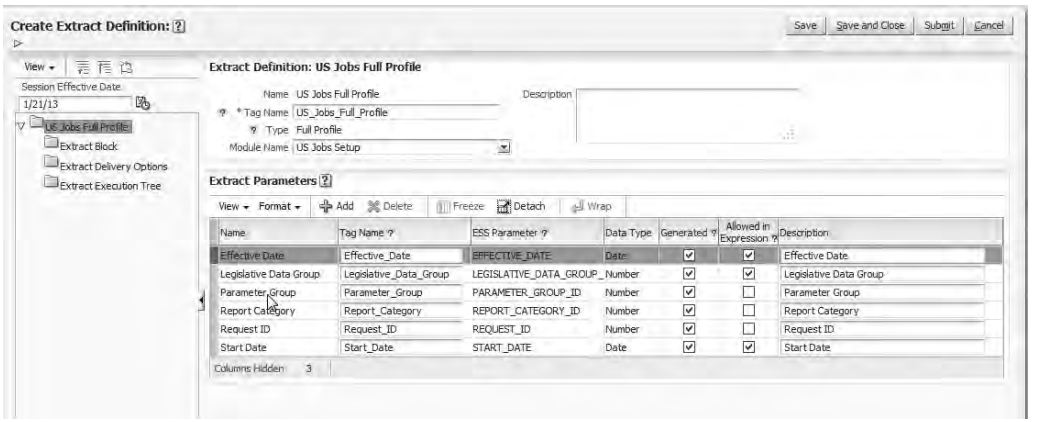


FIGURE 15-8. *HCM export definition*

Inbound Integration Patterns with Fusion Applications

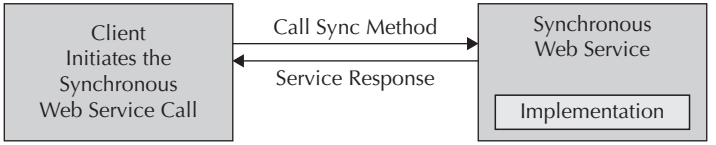
Similar to outbound integration, you may need to do inbound integration with Fusion Applications to bring in data or take certain actions based on the changes happening in your other applications. Most of the products in Fusion Applications support both real-time and batch integration capabilities. We will discuss some of the common integration points in this section for inbound calls to Fusion Applications.

Calling Web Services

Invoking ADF service from a client can be done in various ways. You can use Web service binding or BPEL entity variables if you are using BPEL, or use a proxy class if it is a Java client or any other tool to call a service. You can invoke the service as synchronous request-response invocation or asynchronous request-reply invocation.

Making a Synchronous Service Call

Fusion Applications expose synchronous services. You will use synchronous service calls when you expect the call to return very quickly. These services return the response to the caller, and until the response is returned, the caller will keep waiting as shown in the following illustration.



Making an Asynchronous Service Call

Fusion Applications expose several services that might be long-running or take a significant amount of time for the client or caller to keep waiting for the reply. Fusion ADF Services typically expose asynchronous interfaces in addition to synchronous for a given service operation. You can find the details in OER for a given service. The asynchronous service call can be easily made from a BPEL process, and the process will wait for the response from the service to come back. To provide this asynchronous interface and right callback to waiting clients, Fusion Applications use the Advance Queue (AQ) feature in the Fusion database. The Fusion Applications database has these AQs per product family to handle request and response, such as CRM_AsyncWS_Request and CRM_AsyncWS_Response. The administrator can monitor these queues for diagnostics. The SOA run time uses a Web service addressing mechanism to automatically correlate the asynchronous service call back to the right client. The following list describes the flow of events when you make an asynchronous service call, as shown in Figure 15-9.

1. The client calls an asynchronous method.
2. The asynchronous Web service receives the request and stores it in the request queue.
3. The asynchronous Web service sends a receipt confirmation to the client.

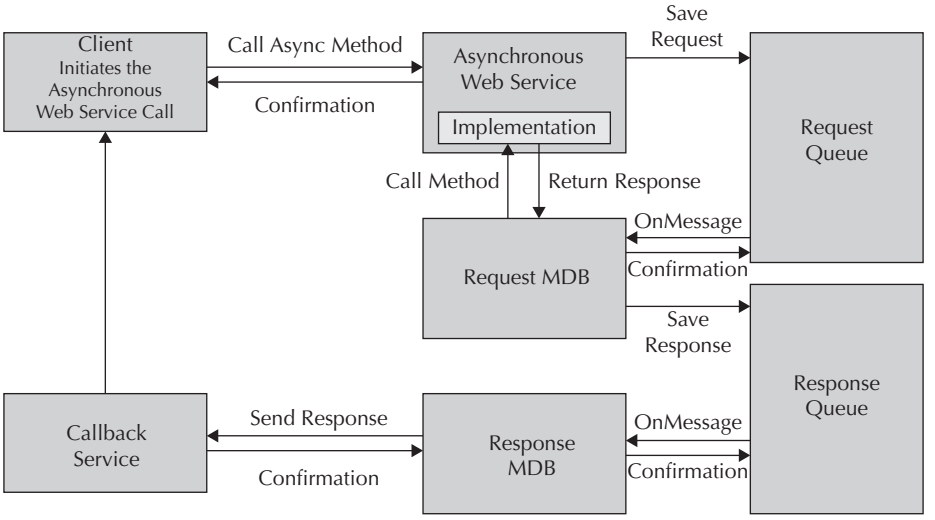


FIGURE 15-9. *Asynchronous service call*

4. The MDB listener on the request queue receives the message and initiates processing of the request.
5. The request MDB calls the required method in the Web service implementation.
6. The Web service implementation returns the response.
7. The request MDB saves the response to the response queue.
8. The request MDB sends a confirmation to the request queue to terminate the process.
9. The onMessage listener on the response queue initiates processing of the response.
10. The response MDB, acting as the callback client, returns the response to the callback service.
11. The callback service delivers the response to the client that initiated the request if the client was waiting for the response.
12. The callback service returns a receipt confirmation message.
13. The response MDB returns a confirmation message to the response queue.

Primary Key Management

When integrating using Web services, the ADF Services typically need the caller to pass the surrogate primary key of the object to identify the data to be updated. Sometimes the services also expose the user-friendly alternative keys in the service interface and the caller can pass those alternative keys to identify rows to be updated instead of surrogate primary key values. In both cases, you will need a way to cross-reference the keys between Fusion Applications and your other external application that you integrate with Fusion Applications. There are four ways in which you can achieve this cross-referencing.

Cross-Reference in Fusion In this approach, Fusion Applications store the cross-reference to external system keys. Some Fusion Applications such as Customer Data Management have a built-in infrastructure to store such cross-references. It simply maps the Fusion surrogate primary key to the external system keys that you will have at hand when you get the event originated in the external system. You can then query Fusion Applications using those external keys to get Fusion surrogate key values to be used for calling the service.

Cross-Reference in External Systems In this approach, the cross-reference to Fusion Applications entities is stored in your external system. It is very similar to the first approach, but the mapping is stored in an external system. When you want to call the Fusion Applications Web service, you will need to look up the cross-reference in your external system to identify the surrogate keys and then use those keys to call the service.

Cross-Reference in Middleware The Fusion Middleware SOA Suite has built-in cross-reference functionality that you can use to manage the cross-reference between your external applications and Fusion Applications. You may choose to use this approach if you want to maintain this cross-reference outside of your applications and be able to use it for many cross-application integrations. You will use this middleware cross-reference map to look up keys before calling necessary services.

Using Alternative Keys Some of the Fusion Applications services allow you to pass alternative business keys in addition to surrogate primary key values. In such cases, when you create the data, you can populate these alternative key attributes in Fusion with exact values from your external or source system. When you need to update the data in future, you can simply pass the value of these keys as you get them from your external system without any lookup. Note that this approach works only for one-to-one system integration. If you had multiple systems with different values for these alternative keys, this approach will not work and you will have to keep the cross-reference for each system by following one of the three options discussed earlier.

Security

Fusion Applications Web services are secured using Oracle Web Service Manager (OWSM), which follows Web service security standards. Different services implement different security policy on the server side. The clients need to use appropriate client-side policy accordingly in order to make a successful service call. You can examine the security policy on a given service either in the WSDL or in the Enterprise Manager where the service is deployed. The administrator can modify the out-of-the-box security on a service using Enterprise Manager. Please read the *Oracle Fusion Middleware Security and Administrator's Guide for Web Services* to understand more about the security and how to call services with different policies.

Bulk Import

Many Fusion Applications provide bulk import capabilities that you can use to do initial loads or migrate data from your legacy or external applications. To bulk-import data, you will first need to load the raw data in the interface tables. Then run the background processes from Fusion Applications to load the interface data into the

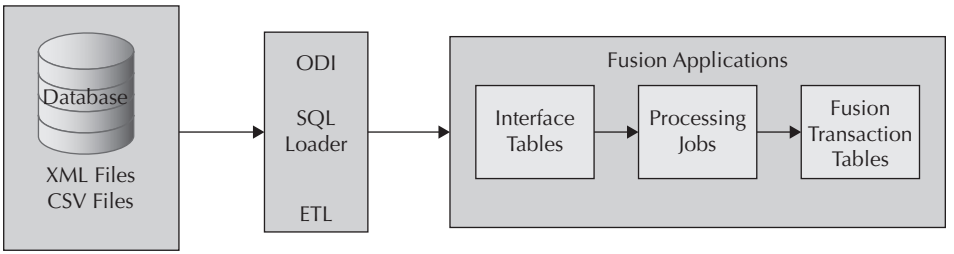


FIGURE 15-10. *Bulk import data flow*

transaction tables. Please read product-specific documentation to understand what capabilities are provided to load specific object data. The typical bulk import flow looks like Figure 15-10.

File Import

Some of the Fusion Applications, such as CRM and HCM, provide utilities to import data from files from end users. The tools take files in formats such as CSV, XML, and XLS as input from the desktop of the user or from a network-accessible location. The tool then uses the regular interface tables and kicks off the import processing. These loaders also provide user interfaces to monitor and review the progress of the process and inspect the errors. You can use the Manage Import Activities task to use a file import interface for CRM applications. You can choose the object from this UI that you want to import and then use the mapping tool to specify what attribute from your input file format maps to the object attribute and submit the batch to start the import process. Please read product-specific documentation to understand more about the specific capabilities of the file import.

An Example Integration Using Standard Patterns

In this section, we will discuss a sample integration using some of the patterns discussed in this chapter. We will use SOA Composite and Web services to do both outbound and inbound integration with Fusion Applications. The high-level integration flow looks like the chart in Figure 15-11. Note that in this example, we are using Fusion Applications as a source of both outbound and inbound integration. The event is originated in Fusion Applications when you create a new location using a page, and you get the outbound message in the form of business event in the SOA

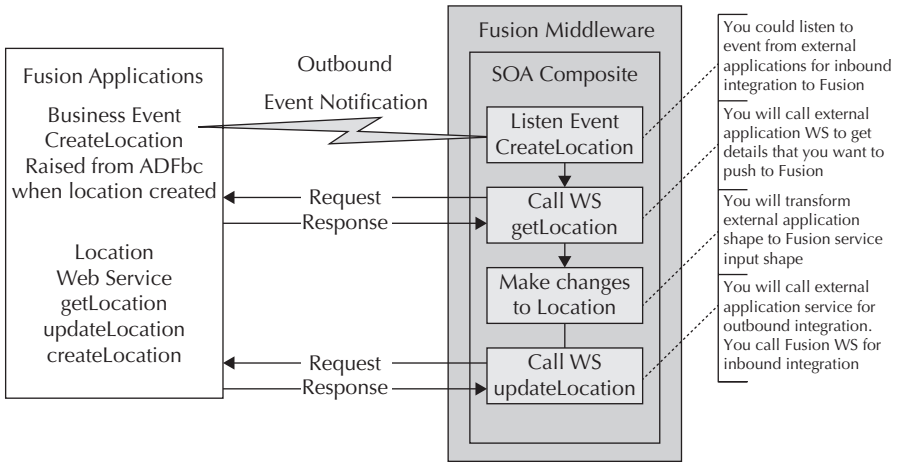


FIGURE 15-11. *Typical integration pattern between applications*

composite. You will then use a synchronous Web service operation to get the details of the location created out of the Fusion Application. And then make an inbound service call to the same Fusion Application to make some changes to the location. In real integration scenarios, the application on the other side will be your external application that you want to integrate with Fusion Applications.

To build this integration, we will need to follow several sets of tasks.

Define a Connection to MDS Repository to Find an Event Definition

Once you have identified the name of the business event that you need to listen to, either using OER or from product documentation, you can discover the event definition and schema using the SOA MDS repository on your environment.

1. Go to JDeveloper and view the Resource Palette. Select New Connection: SOA MDS as option to create a new connection to your Fusion Applications SOA MDS repository.

2. Give your connection a name, select Connection Type as DB Based MDS, provide database connection details for the CRM_FUSION_MDS_SOA schema user, select MDS partition as soa-infra, and test that the connection is successful.

3. Once the connection is created, you can expand the connection and find out the event definition for the create location at oracle.apps.cdm.foundation.parties.publicModel.locations.entity.events as shown in Figure 15-12.

FIGURE 15-12. *Finding a location event in MDS connection*

Define a New SOA Composite Application and Subscribe to Events

In this section, we review the steps to define a new SOA composite and subscribe to the event that is raised when a new location is created.

1. Select the New Application option, give it the name **FusionAppsLocationIntegration**, and choose the SOA Application template as shown in Figure 15-13.
2. Click Next and name the project as **LocationIntegration**. Click Next, give the composite name **LocationIntegration**, choose template Composite With Mediator as shown in Figure 15-14, and click Finish to complete the wizard.
3. In the Create Mediator dialog, give it the name LocationCreate and choose the template Subscribe to Events. Now click the Add icon to subscribe to a new event as shown in Figure 15-15.

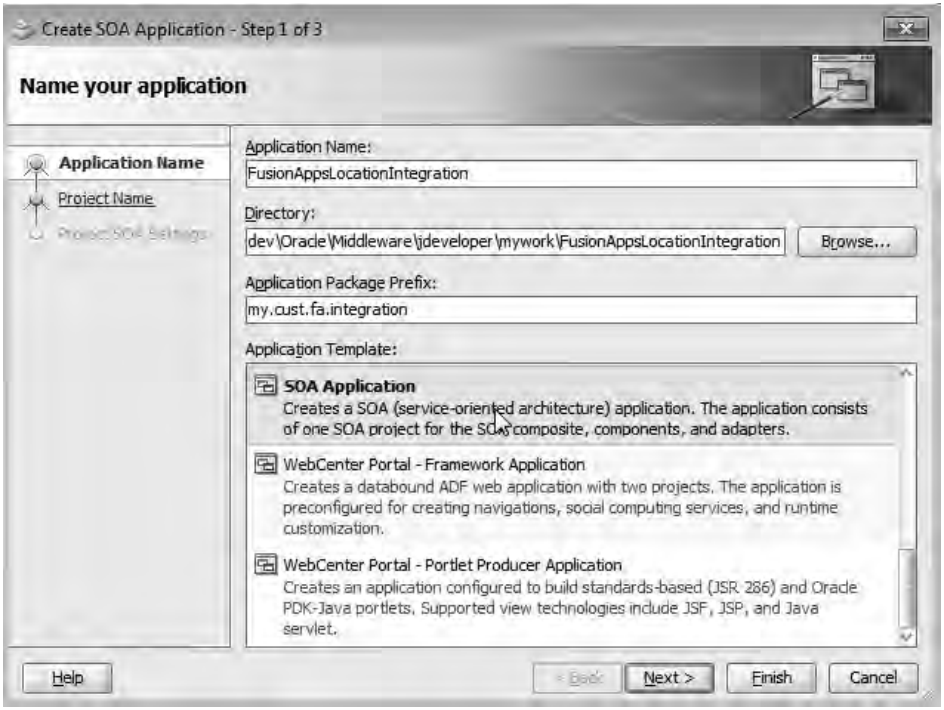


FIGURE 15-13. *Creating a new application*

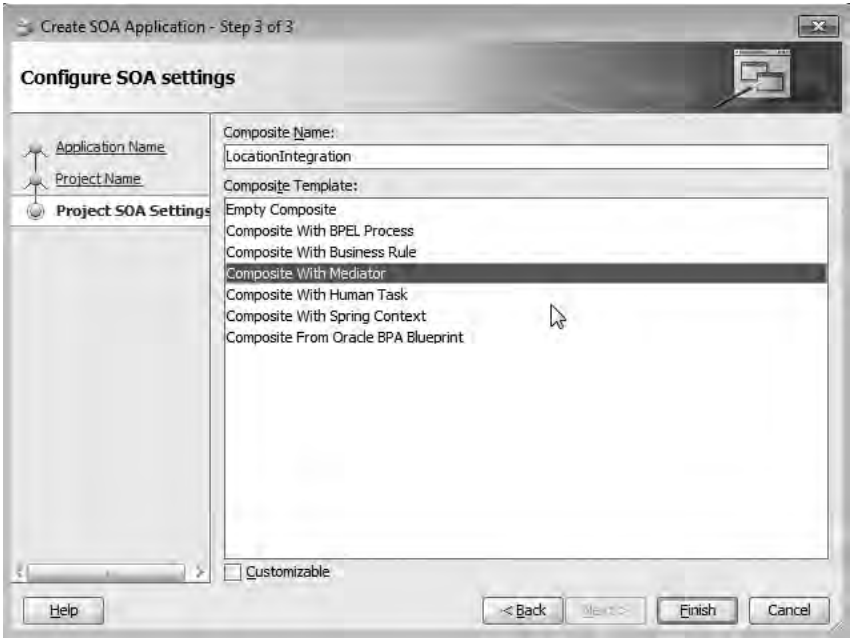


FIGURE 15-14. *Creating a new SOA Composite With Mediator project*

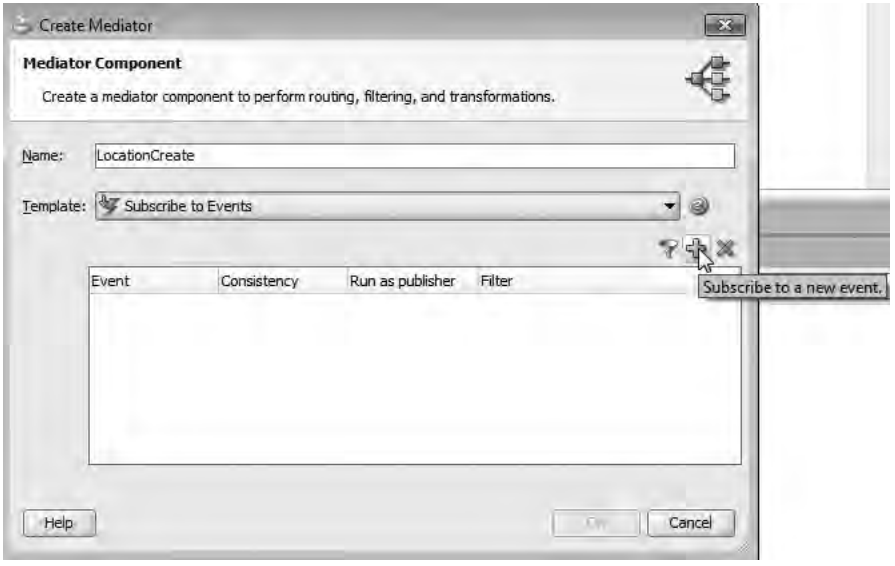
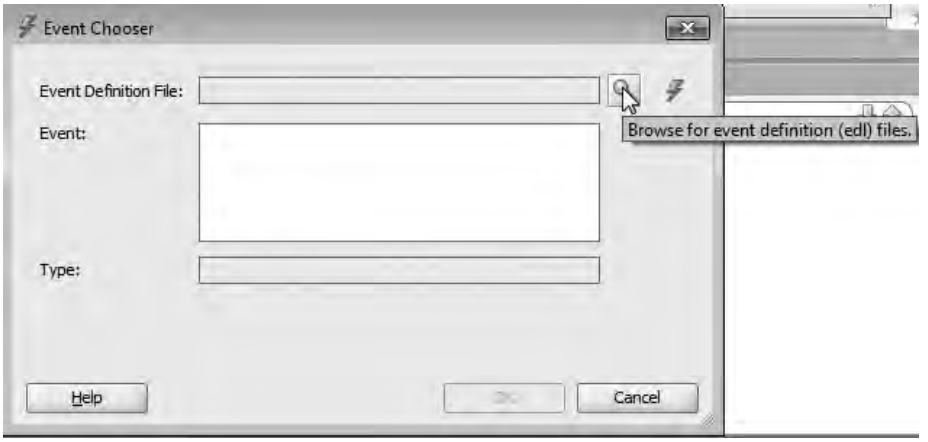
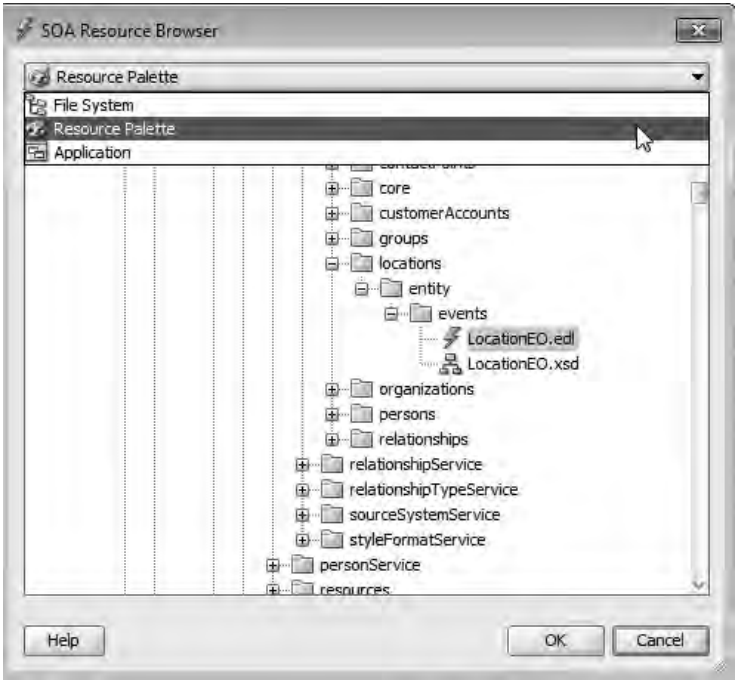


FIGURE 15-15. *Subscribe to an event using Mediator*

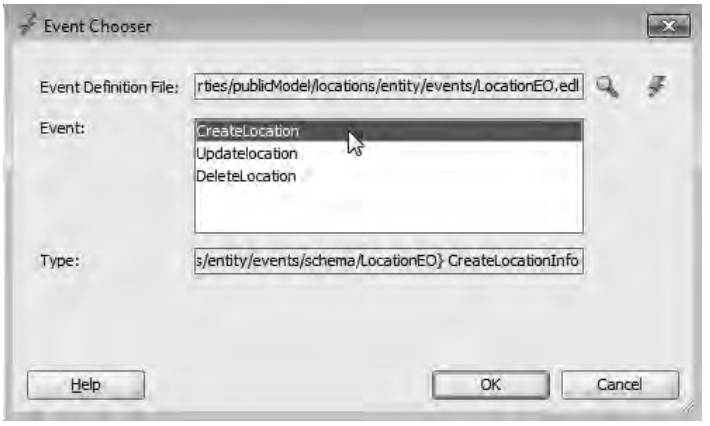
4. Click the browse icon for picking the event definition file for the Location Creation event.



5. In the SOA Resource Browser dialog, select Resource Palette and select LocationEO.edl.



6. Choose CreateLocation as the event name in the Event Chooser dialog and click OK to complete the wizard.



Create a BPEL Process and Route the Event to the Process

In this section, we define a new BPEL process that will orchestrate our flow for integration and route the incoming event from the SOA composite to the BPEL process.

1. Select File | New from the menu and choose to create a new BPEL Process as shown in Figure 15-16.
2. Name the BPEL process as ProcessLocation. Choose the Synchronous BPEL Process template and set the transaction to requiresNew as shown in Figure 15-17.

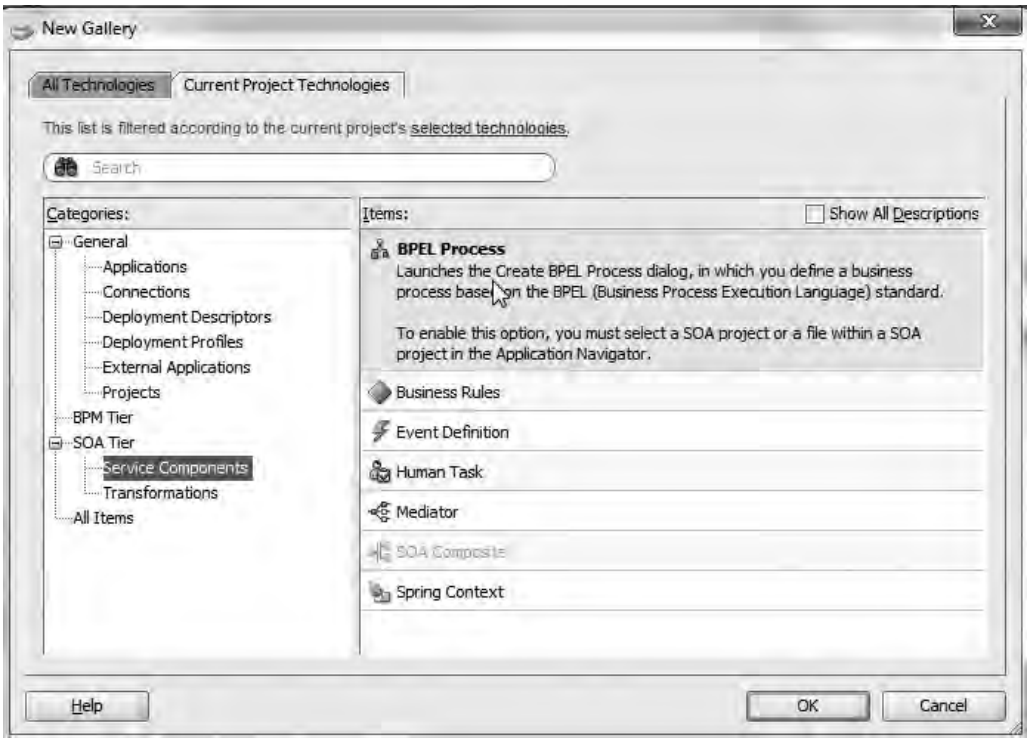
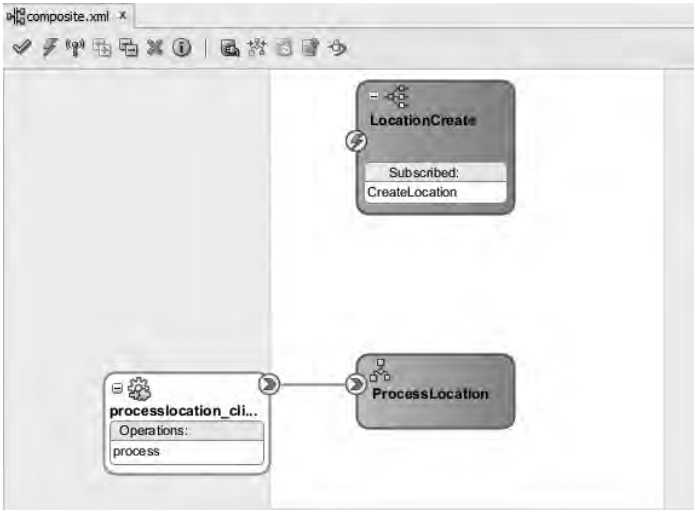


FIGURE 15-16. *Creating a new BPEL process*

3. This illustration shows the SOA composite.



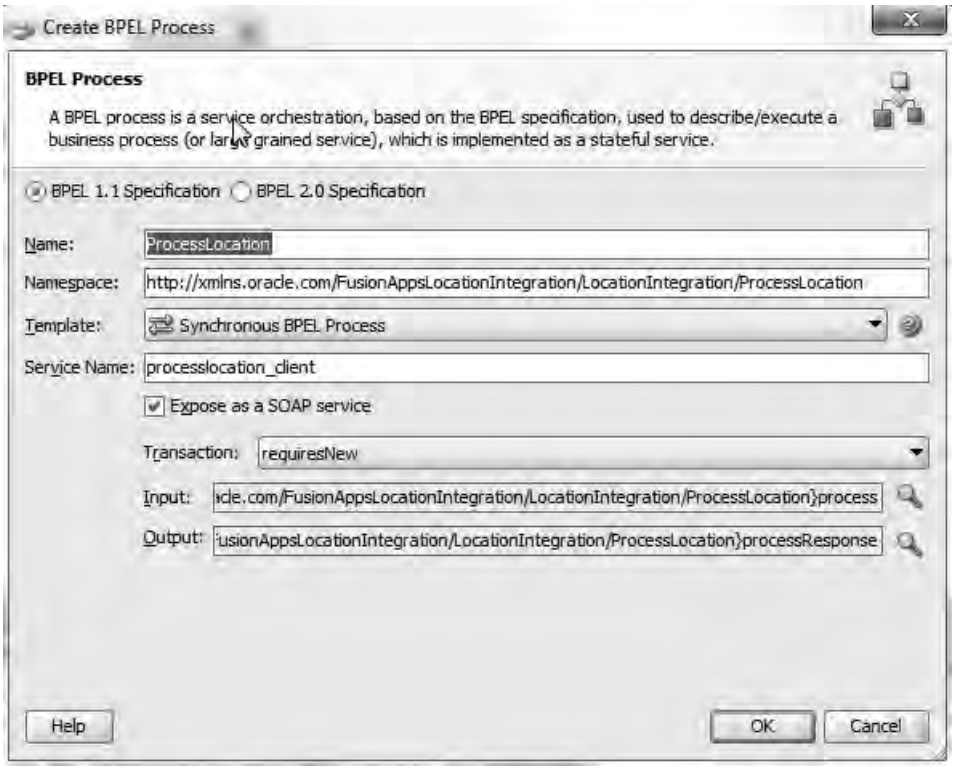
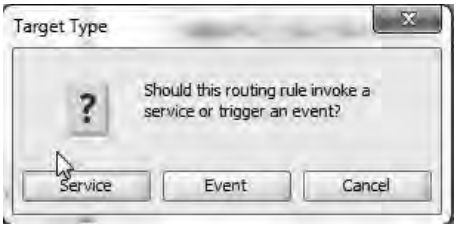


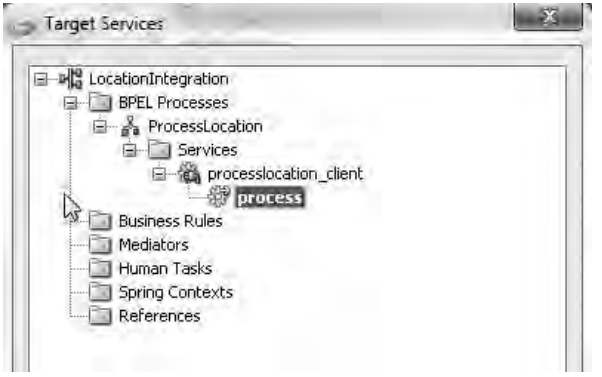
FIGURE 15-17. *Configure BPEL process properties.*

4. Now double-click on the LocationCreate mediator. Click the green add icon (+) to define a new routing rule that will route the event to a BPEL process as shown here.

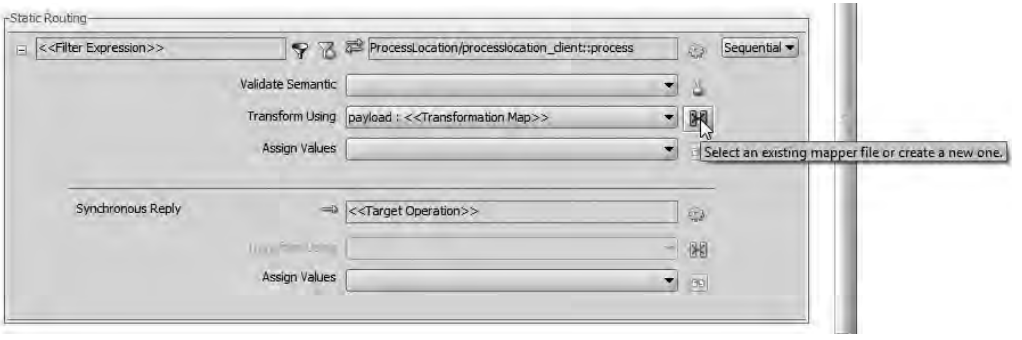
5. Select Service as the routing rule target type.



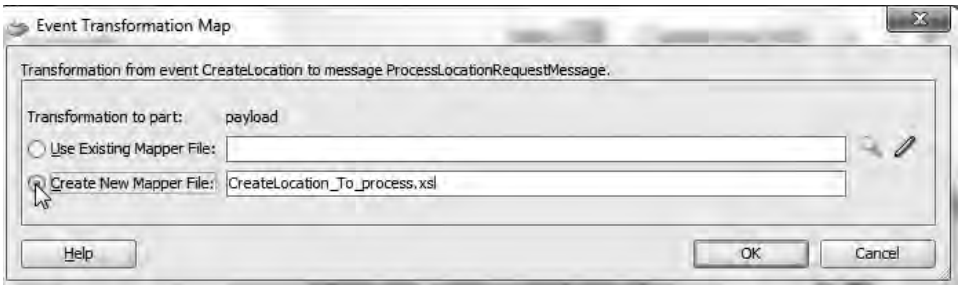
6. Choose the newly defined BPEL process as the target service.



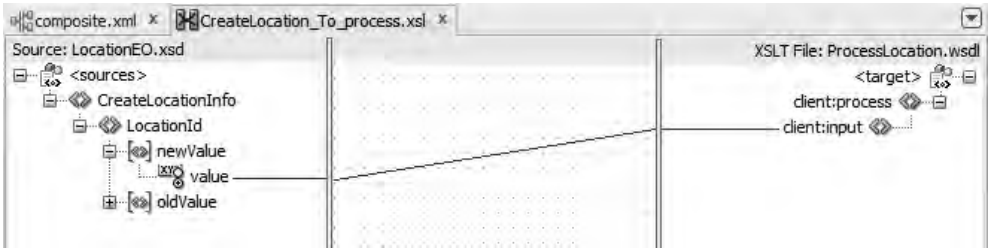
7. Click the Transform icon to pass the LocationId value from the event to the BPEL process.



8. Choose the Create New Mapper File option in the Event Transformation Map dialog.



9. On the XSLT transformation, map the LocationId : newValue : value field to the BPEL process input field by drag and drop.



10. The SOA composite after this routing rule looks like Figure 15-18.

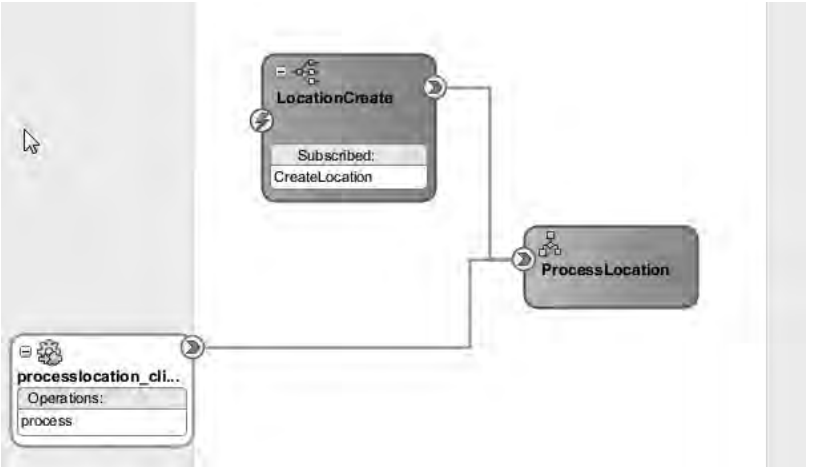
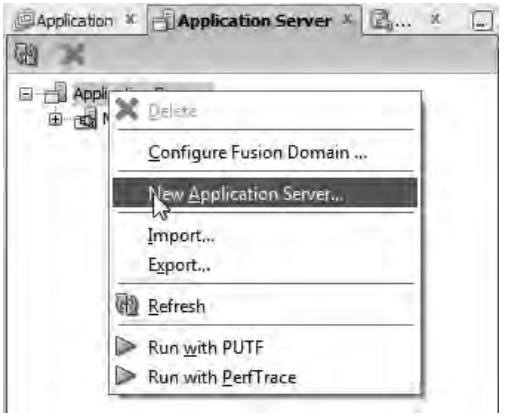


FIGURE 15-18. *SOA Composite With Mediator routing to BPEL process*

Deploy to Application Server and Test Event Subscription

Now we will deploy this SOA composite to an application server for testing.

1. Select Application Server Navigator from the View menu. Right-click and choose the New Application Server option as shown in the following illustration. Complete the wizard by providing the server name, port, and login credentials, and then test the connection.



2. Right-click on the LocationIntegration project and choose the Deploy: LocationIntegration option. In the Deployment Action dialog, choose the Deploy to Application Server option as shown in Figure 15-19. Choose the application server connection created in Step 1 and complete the wizard to deploy the SOA composite to the server.

3. You can use the Customer Center application in CRM or Receivables application in Financials or Party Center from Customer Data Management to create a new party along with address. This will create a new location and raise the CreateLocation event that our SOA composite subscribes to. Figure 15-20 shows the Create Organization page in party center.

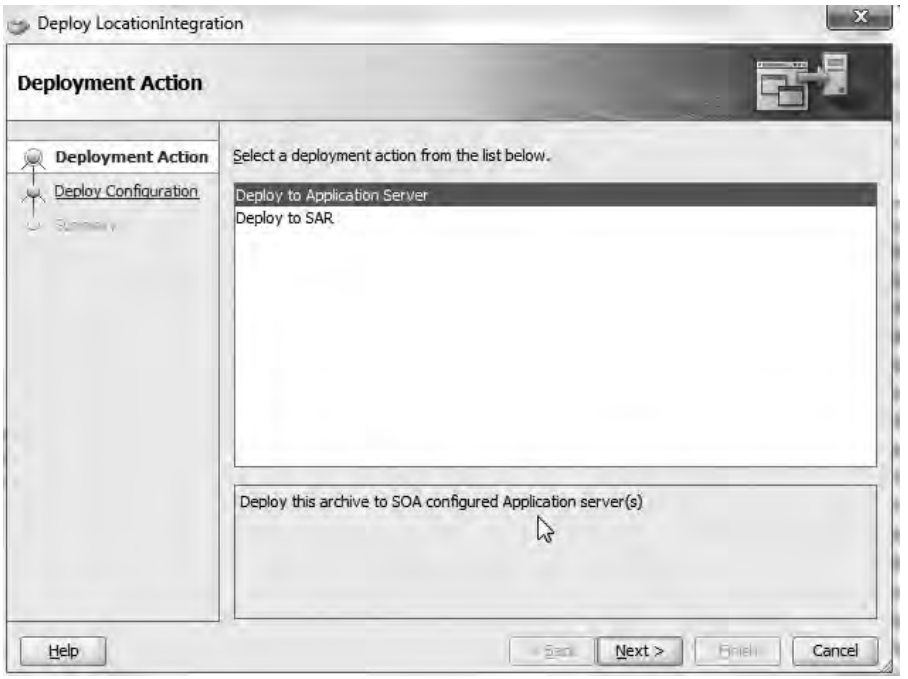


FIGURE 15-19. *Deploying SOA composite*

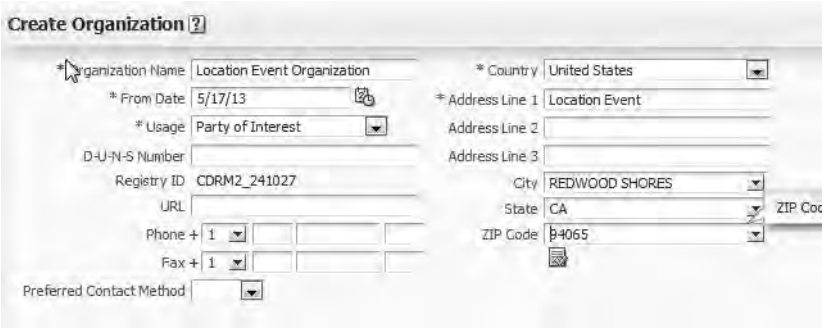


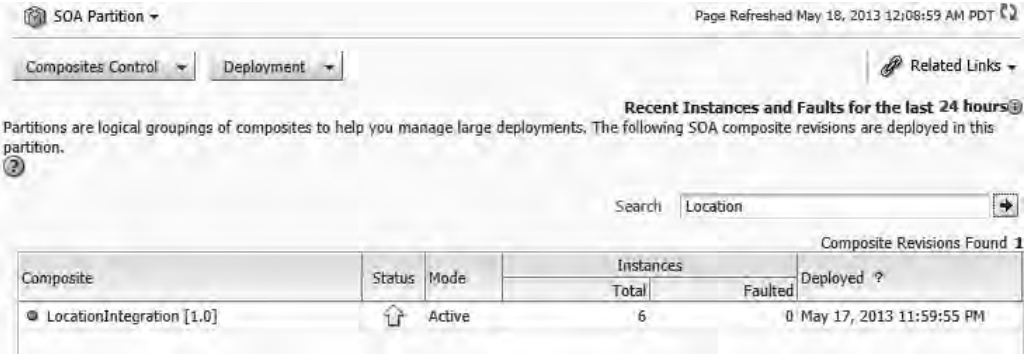
FIGURE 15-20. *Create Organization application user interface*

```
            <parameter>
               <name>USER_DISPLAY_NAME</name>
               <namespace>FND$SECURITY</namespace>
               <value>Hunter Robinson</value>
            </parameter>
         </fmw-context>
      </context>
      <tracking>
         <ecid>29ea7a02aaeed5a2:61b22679:13eb4f9342c:-8000-000000000000d5e9</ecid>
      </tracking>
      <content>
         <CreateLocationInfo xmlns="/oracle/apps/cdm/foundation/parties/publicModel/locations/entity/events/schema/LocationEO">
            <LocationId>
               <newValue value="300100020683800"/>
            </LocationId>
         </CreateLocationInfo>
      </content>
   </business-event>
   [17-MAY-2013 10:30:15]
```

FIGURE 15-21. *Business event in EDN log*

4. You can check if a given business event was raised or not by going to EDN logs for your SOA deployment at http://host:port/soa-infra/events/edn-db-log. Once you save the data on UI, you can see the CreateLocation event in the soa edn log as shown in Figure 15-21.

5. Now we will check the SOA composite instance and validate whether the event was delivered and our BPEL process was triggered or not. Log in to the Enterprise Manager at http://host:port/em. Once you log in to em, expand SOA : soa-infra and click Default. Search for Location to find your SOA composite as shown in the following illustration.



6. Click on LocationIntegration composite to check all instances of this SOA composite. Click on the first Instance ID as shown in the following illustration to see details.

7. This launches the composite Flow Trace window. You can see that the composite received the CreateLocation event that was received by LocationCreate mediator and routed to ProcessLocation BPEL process as shown in Figure 15-22.

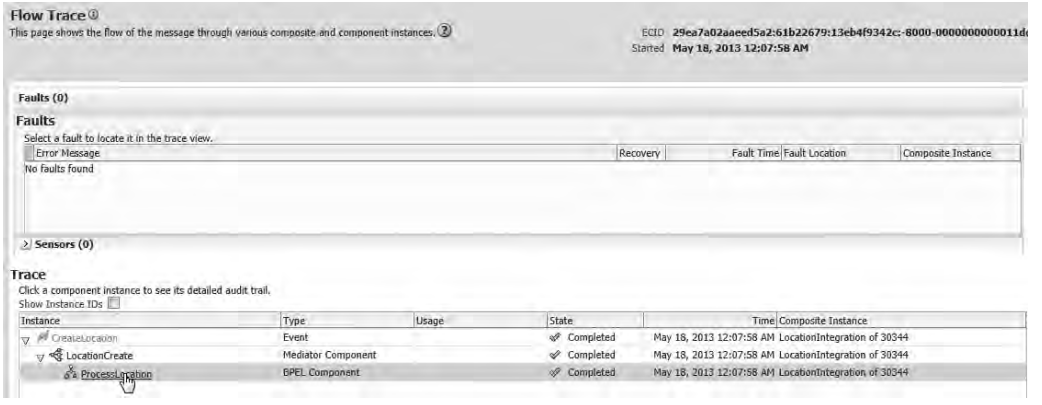


FIGURE 15-22. *SOA composite instance flow trace*

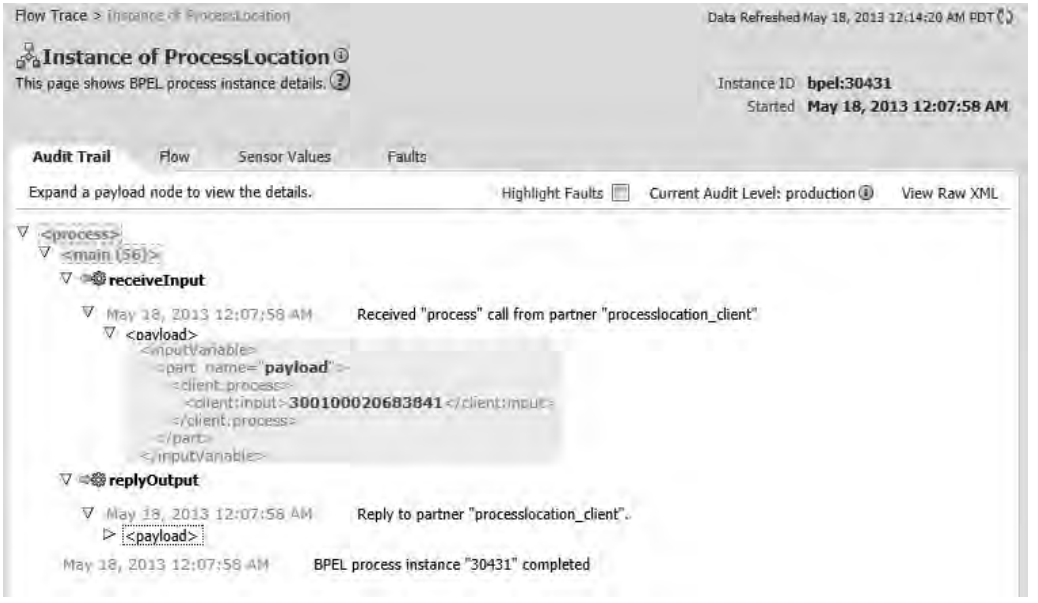


FIGURE 15-23. *Empty BPEL process instance audit trail.*

8. This opens the BPEL process Audit Trail window. As you can see, it shows that the BPEL process received a LocationId that we transformed from the event to BPEL process input as shown in Figure 15-23. We will use this LocationId value in the rest of the BPEL process to now interact with Web services.

Build BPEL Flow to Process Location Data

Now that we have an SOA composite that successfully instantiates on a business event, we will add processing logic in the BPEL. We will first use the getLocation service to get the full details of the location and then make an update and call updateLocation service to make changes and validate that using the UI.

1. Open the BPEL process in the editor. Drag and drop Partner Link from the Component Palette onto the BPEL process as shown in Figure 15-24.

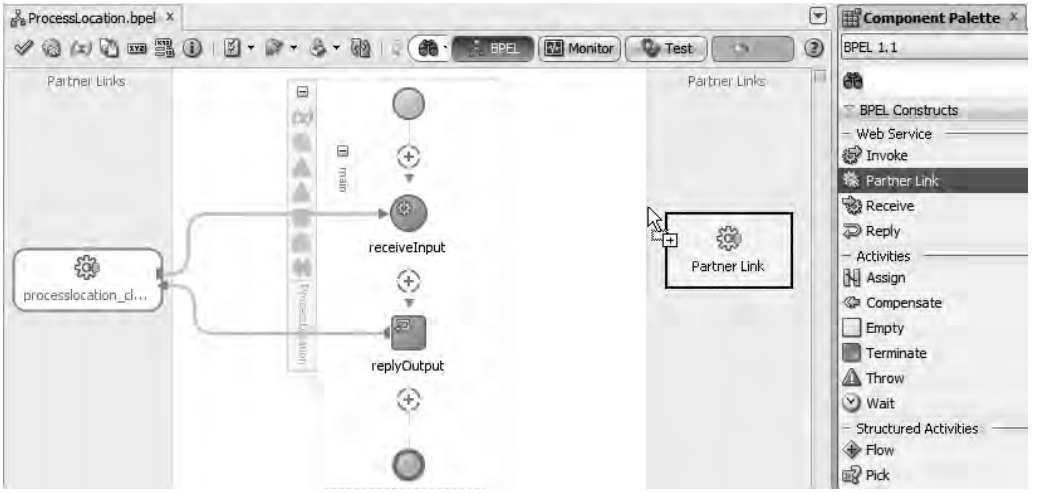
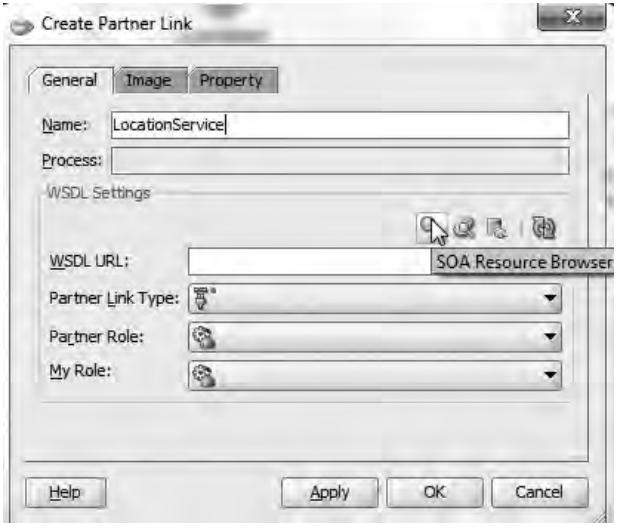
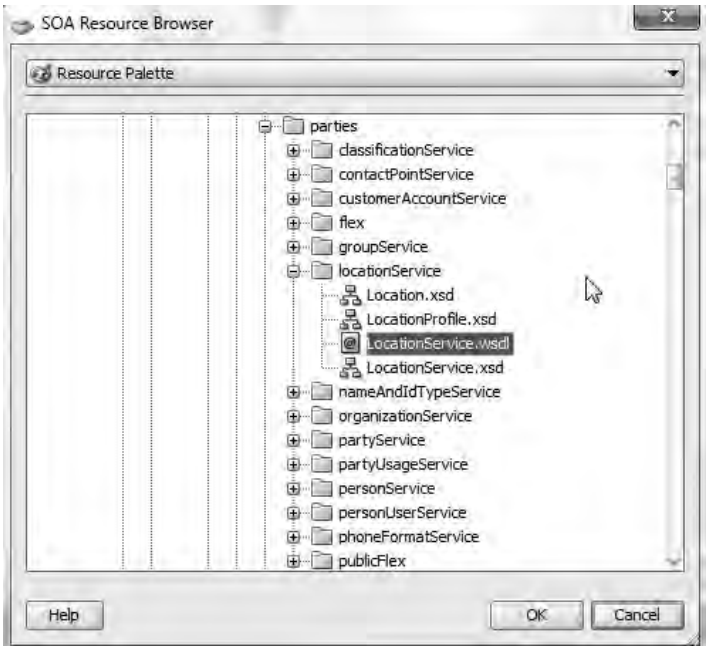


FIGURE 15-24. *Adding a partner link to BPEL process*

2. Give it the name **LocationService** and click on SOA Resource Browser as shown here.



3. In the browser window, select Resource Palette and browse your SOA MDS connection to select oracle.apps.cdm.foundation.parties.locationService.



4. Choose the Partner Link Type and My Role as shown in the following illustration and click OK.

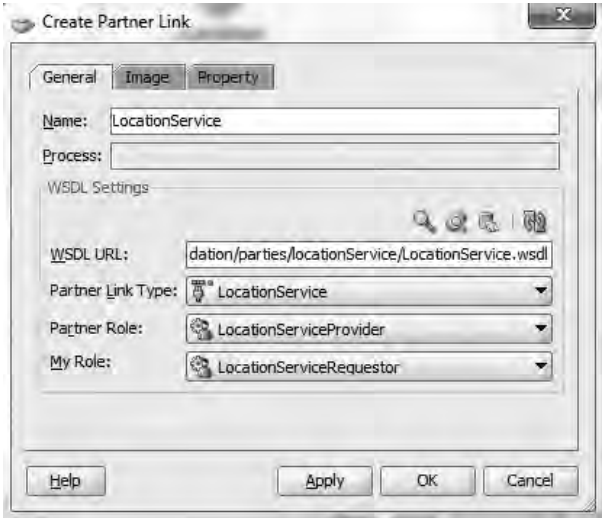


FIGURE 15-25. *Specify namespace in the adf-config file.*

5. Modify the adf-config file from the Application Resources: Descriptors: ADF META-INF folder to use namespace /apps instead of /apps/oracle as shown in Figure 15-25.

6. Now drag and drop the Invoke activity onto the BPEL process between receiveInput and replyOutput. Connect the Invoke activity to LocationService as shown in Figure 15-26.

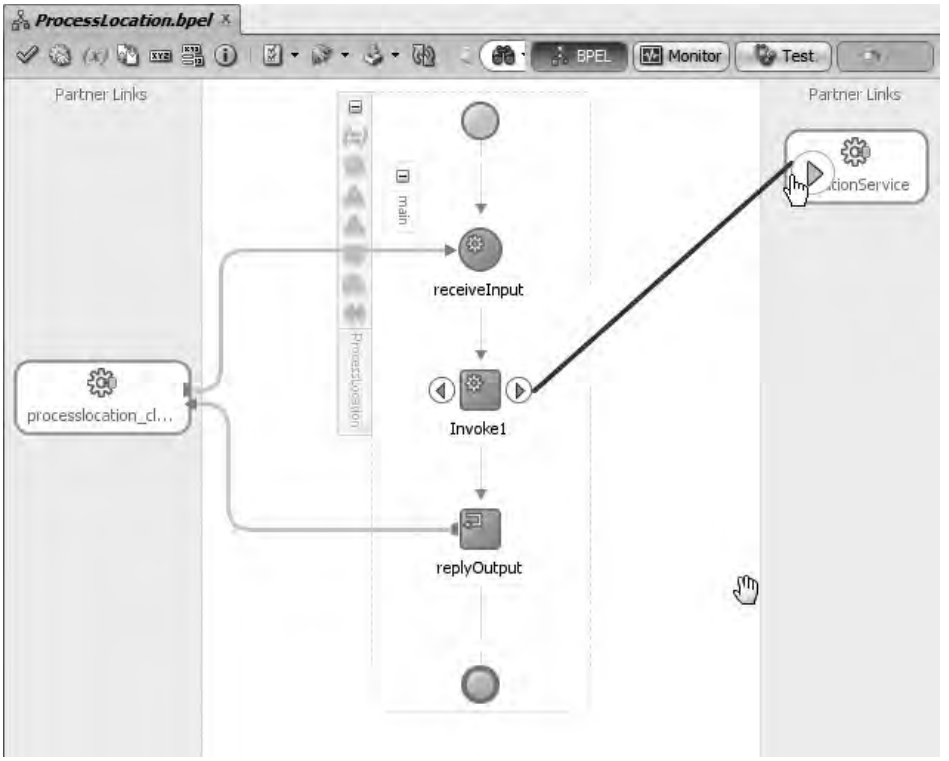


FIGURE 15-26. *Adding Invoke activity for get location service*

7. In the Edit Invoke dialog, provide the name **InvokeGetLocation**. Select getLocation as the Operation. Click the green plus icon to automatically generate input and output variables that will be used for this service call as shown in Figure 15-27.

8. The getLocation service needs the LocationId input parameter. Our BPEL process gets the LocationId as input. We will pass the input of BPEL to the input of the WS invoke variable. Drag and drop the Assign activity on the BPEL process between the receiveInput and InvokeGetLocation activities. Name this activity as AssignLocationId. Double-click on the assign activity and go to the Copy Rules tab. Map the inputVariable input to InvokeGetLocation_getLocation_InputVariable locationId variable as shown in Figure 15-28.

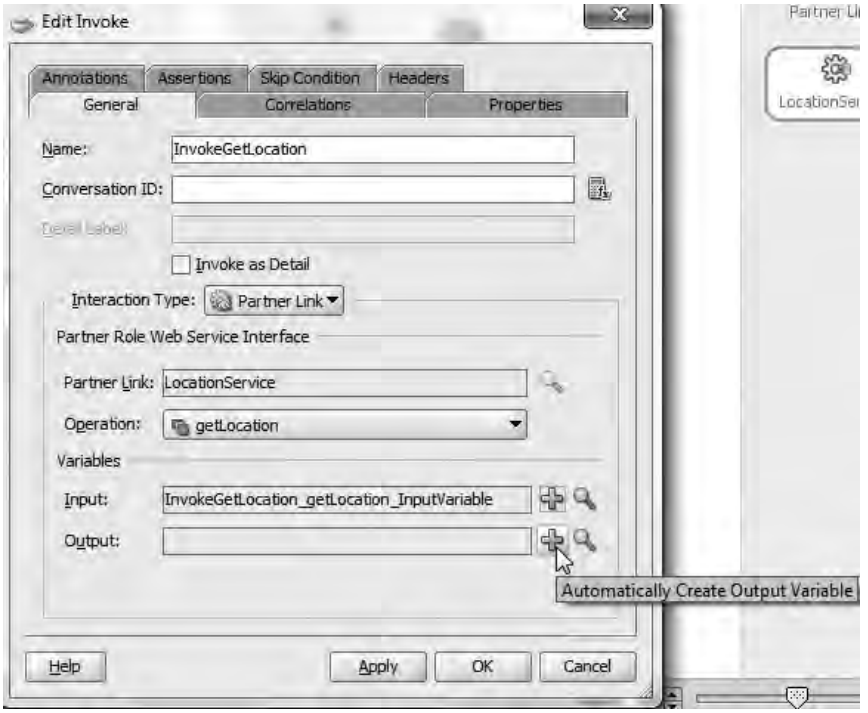


FIGURE 15-27. *Defining operation to invoke and variables*

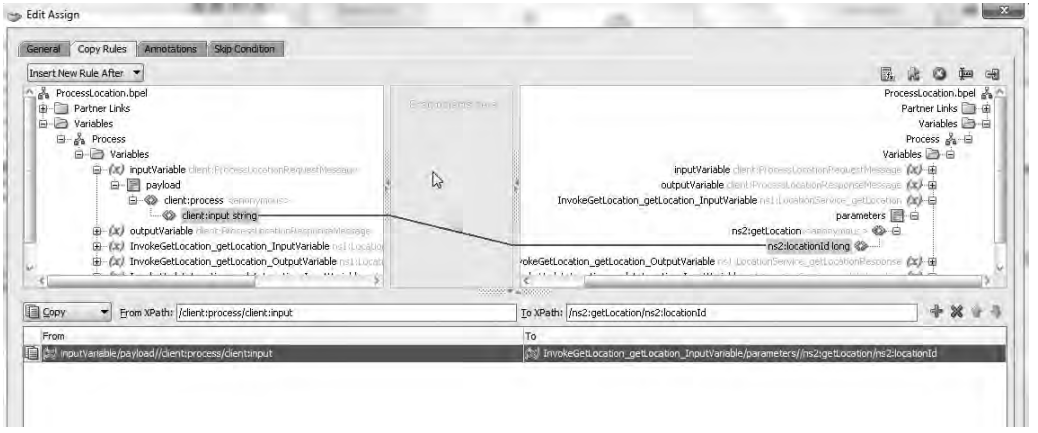


FIGURE 15-28. *Adding Assign activity to pass input variable to a service call*

9. Drag and drop the invoke activity between InvokeGetLocation and replyOutput activity and name it InvokeUpdateLocation. Connect this invoke with the LocationService Partner link, choose updateLocation as the Operation, and choose the default variables as shown in Figure 15-29.

10. Now we will assign the output of the getLocation service call to the input of the updateLocation service call. Later we will modify this to make some updates as well. Drag and drop the assign activity between InvokeGetLocation and InvokeUpdateLocation activity and name it **AssignUpdateLocation**. Map the InvokeGetLocation_getLocation_OutputVariable result to the InvokeUpdateLocation_updateLocation_InputVariable location. Note that both of the variable parts are of same type, Location, and we can map them as shown in Figure 15-30.

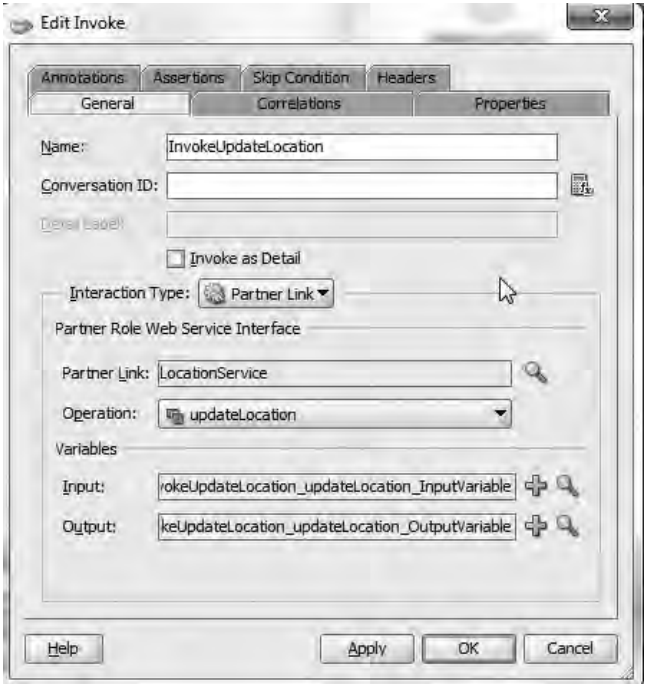


FIGURE 15-29. *Adding Invoke activity for update location service*

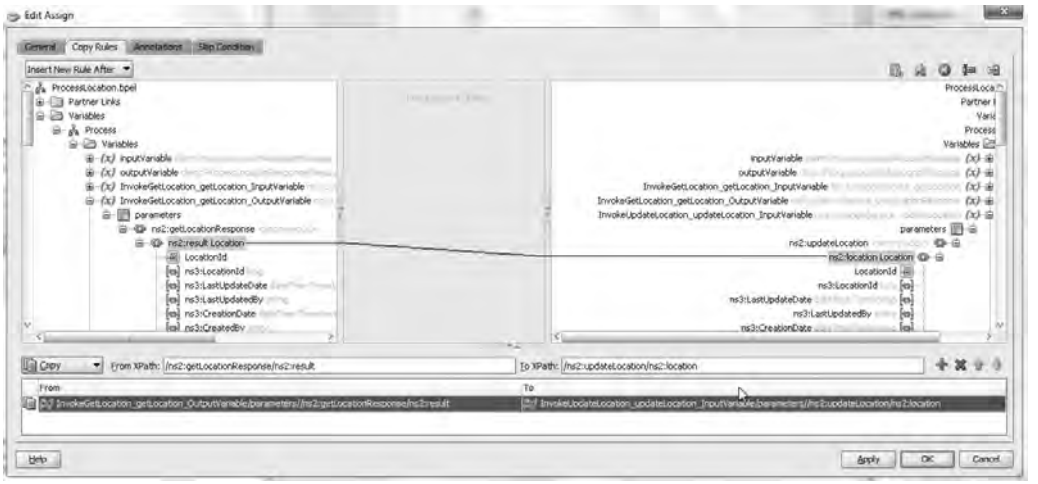
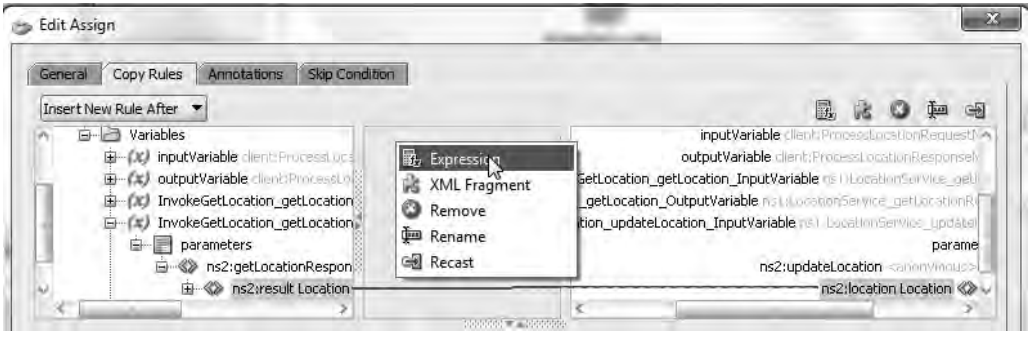


FIGURE 15-30. *Add Assign activity for update location input variable.*

11. Now we will modify address line 1. Right-click on the middle section and choose an expression as shown in the following illustration. Type **Update Address 1** and click OK.



12. Connect this expression to Address1 in the variable InvokeUpdateLocation_updateLocation_InputVariable as shown in Figure 15-31.

13. Now we will need to add appropriate security policies to the service that we are calling from the composite. You can find supported security policies for a given service from OER. Right-click on the service reference in your composite.xml file and choose Configure WS Policies | For Request as

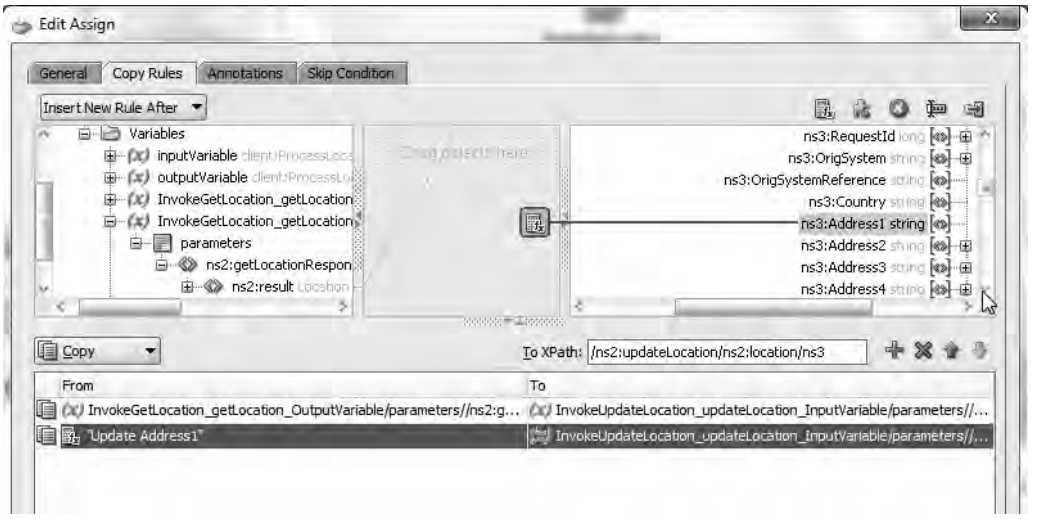
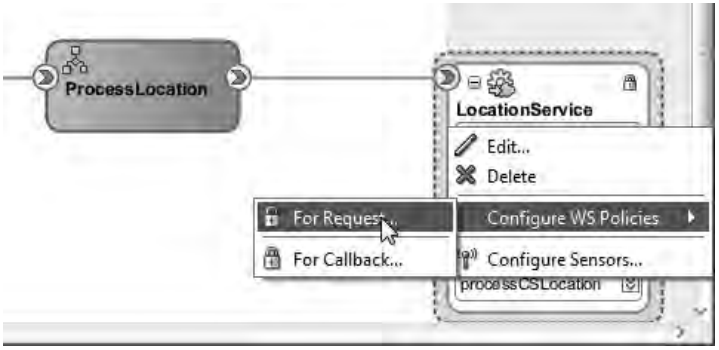


FIGURE 15-31. *Assign the modified value to update the location input variable.*

shown in the following illustration. Click the Add Security Policies button under the Security section and choose the security policy oracle/wss11_saml_token_with_message_protection_client_policy.



14. Similarly, configure Callback security policy and choose oracle//wss11_saml_or_username_token_with_message_protection_service_policy.

15. Validate that you have a service reference in the composite.xml file as shown in the following code. If it is missing, please add manually.

```
  <reference name="LocationService"
           ui:wsdlLocation="oramds:/apps/oracle/apps/cdm/foundation/parties/
locationService/LocationService.wsdl">
    <interface.wsdl interface="http://xmlns.oracle.com/apps/cdm/foundation/parties/
locationService/applicationModule/#wsdl.interface(LocationService)"
                    callbackInterface="http://xmlns.oracle.com/apps/cdm/foundation/
parties/
locationService/applicationModule/
#wsdl.interface(LocationServiceResponse)"/>
    <binding.ws port="http://xmlns.oracle.com/apps/cdm/foundation/parties/
locationService/applicationModule/
#wsdl.endpoint(LocationService/LocationServiceSoapHttpPort)"
    location="oramds:/apps/oracle/apps/cdm/foundation/parties/
locationService/LocationService.wsdl">
      <wsp:PolicyReference URI="oracle/wss11_saml_token_with_message_
protection_client_policy"
                           orawsp:category="security" orawsp:status="enabled"/>
    </binding.ws>
    <callback>
      <binding.ws port="http://xmlns.oracle.com/apps/cdm/foundation/parties/
locationService/applicationModule/
#wsdl.endpoint(LocationService/LocationServiceResponse_pt)">
        <wsp:PolicyReference URI="oracle//wss11_saml_or_username_token_with_message_
protection_service_policy"
                             orawsp:category="security"
                             orawsp:status="enabled"/>
      </binding.ws>
    </callback>
  </reference>
```

16. Make sure the following wiring information is in the composite.xml file for this service reference to BPEL process as shown here.

```
    <wire>
      <source.uri>processlocation_client_ep</source.uri>
      <target.uri>ProcessLocation/processlocation_client</target.uri>
    </wire>
    <wire>
      <source.uri>LocationCreate/ProcessLocation.processlocation_client</source.uri>
```

```
    <target.uri>ProcessLocation/processlocation_client</target.uri>
  </wire>
  <wire>
    <source.uri>ProcessLocation/LocationService</source.uri>
    <target.uri>LocationService</target.uri>
  </wire>
```

Deploy the SOA Composite with Concrete Service URL

In this section, we will deploy our SOA composite such that the service reference in the composite points to the right service URL.

1. The SOA composite service reference points to the WSDL location in the MDS connection we used. We need to use a configuration plan so that this reference service location gets updated with a concrete service URL during deployment. Save the following content as an FADeployPlan.xml file. Replace the host and port in the file to point to the real host and port where your service is deployed.

```
<?xml version="1.0" encoding="UTF-8"?>
<SOAConfigPlan xmlns="http://schemas.oracle.com/soa/configplan">
  <composite name="LocationIntegration">
    <reference name="LocationService">
      <binding type="ws">
        <attribute name="location">
          <!--replacing node for portType {http://xmlns.oracle.com/apps/cdm/
foundation/parties/
locationService/applicationModule/}LocationService-->
<replace>http://host:port/foundationParties/LocationService?WSDL</replace>
        </attribute>
      </binding>
    </reference>
  </composite>
</SOAConfigPlan>
```

2. Select the SOA composite project and build it.
3. Right-click on the project and select LocationIntegration.
4. Select Deploy to Application Server from Deployment Action.
5. Choose this file as deployment plan as shown in Figure 15-32.
6. Choose your application server in the next step and complete the deployment wizard.

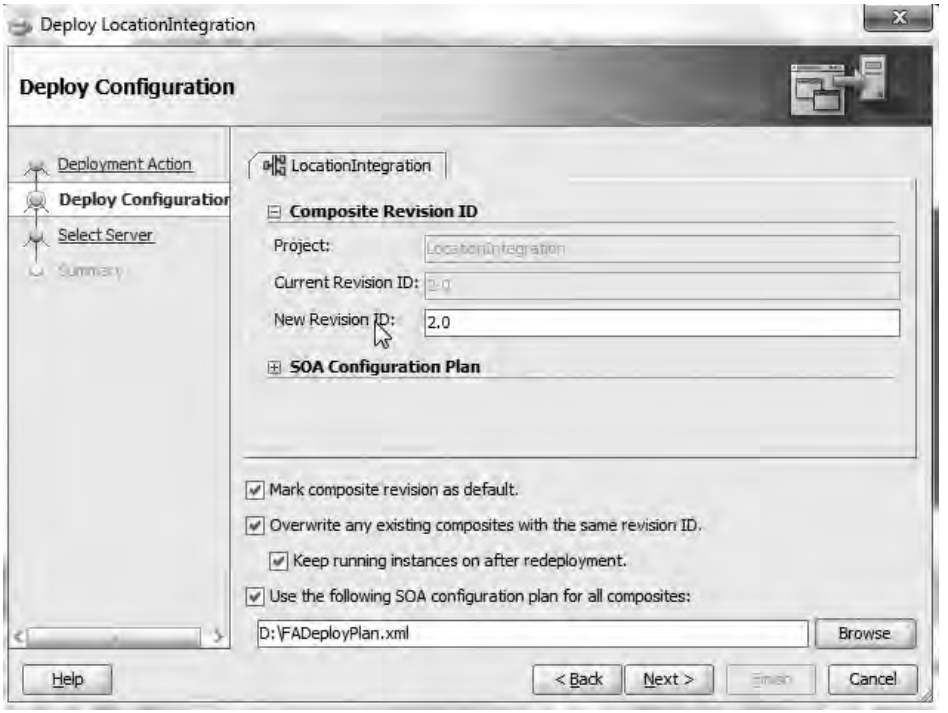


FIGURE 15-32. *Using SOA configuration plan for deployment*

7. Go to Enterprise Manager for your deployment at http://host:port/em and navigate to SOA: soa-infra: default and click on your SOA composite LocationIntegration.

8. Click on the SOA Composite drop-down and select the Export option.

9. On the Export Composite page, choose Option 1: Export with All Post-deploy Changes and click the Export button. This downloads the SOA jar file. Unzip the file, open the composite.xml file, and validate that the service reference is updated to a concrete URL instead of the MDS URL you see in JDeveloper: location="oramds:/apps/oracle/apps/cdm/foundation/parties/locationService/LocationService.wsdl". If you see the right URL, it confirms that the deployment was accurate.

Test the Complete Integration Flow

We will test the complete integration flow now and see how to validate.

1. Use any application to create a new address as indicated previously.
2. Go to EM and find the instance for your SOA composite. Click on the instance ID to open the composite to see the details of our flow. Expand the trace and you can see that the composite listens to the event and calls the BPEL process, and the BPEL process has successfully called the two services, as shown in the following illustration.

3. Click the ProcessLocation link to open the BPEL process and examine the Audit Trail.
4. You can expand each step in the flow and see the data being processed as shown in Figure 15-33.

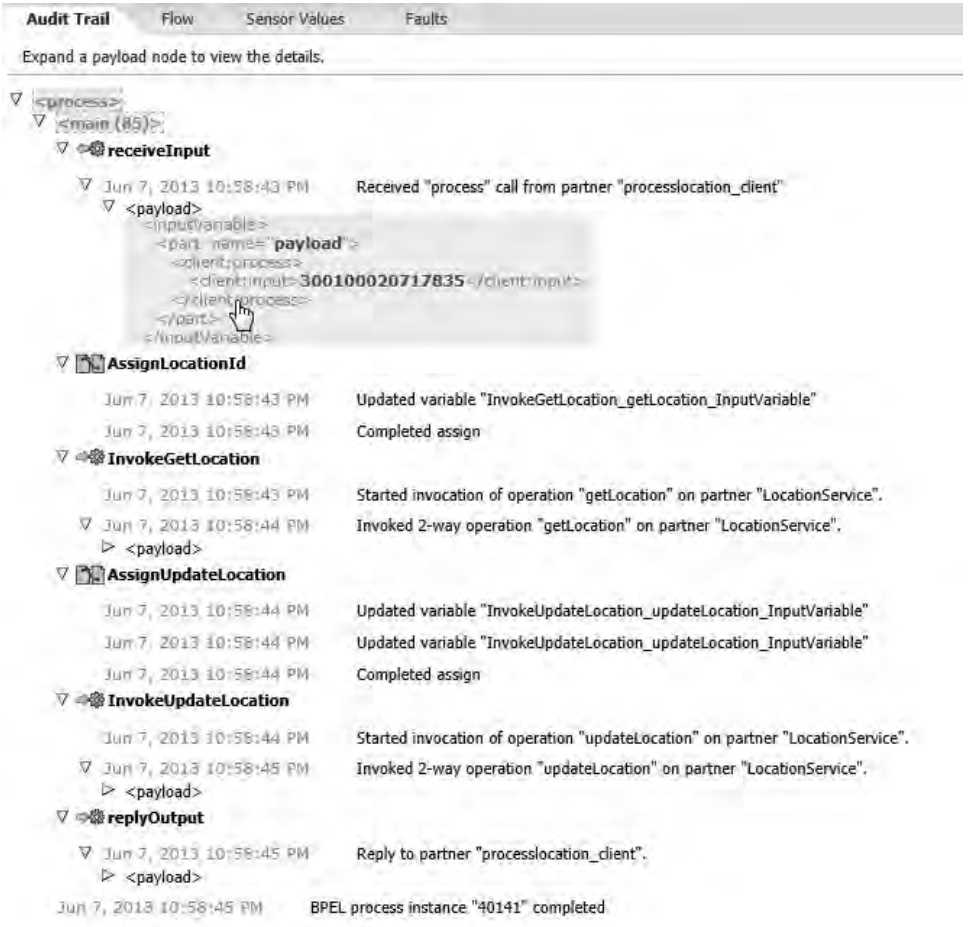


FIGURE 15-33. *Examine the BPEL process audit trail.*

How to Test Fusion Applications Web Services

You can test Fusion Applications Web Services using Enterprise Manager. You can use OER to identify what role is needed to access a given service. Once you identify the role, you can then create a user and assign that particular role to that user and invoke the service with that user for testing. For our example, we will need a user

who has the Master Data Management Application Administrator role, which gives access to Location Service.

1. Log in to EM where your service is deployed.
2. Right-click on the server that hosts your service and select the Web Services menu option as shown in Figure 15-34.

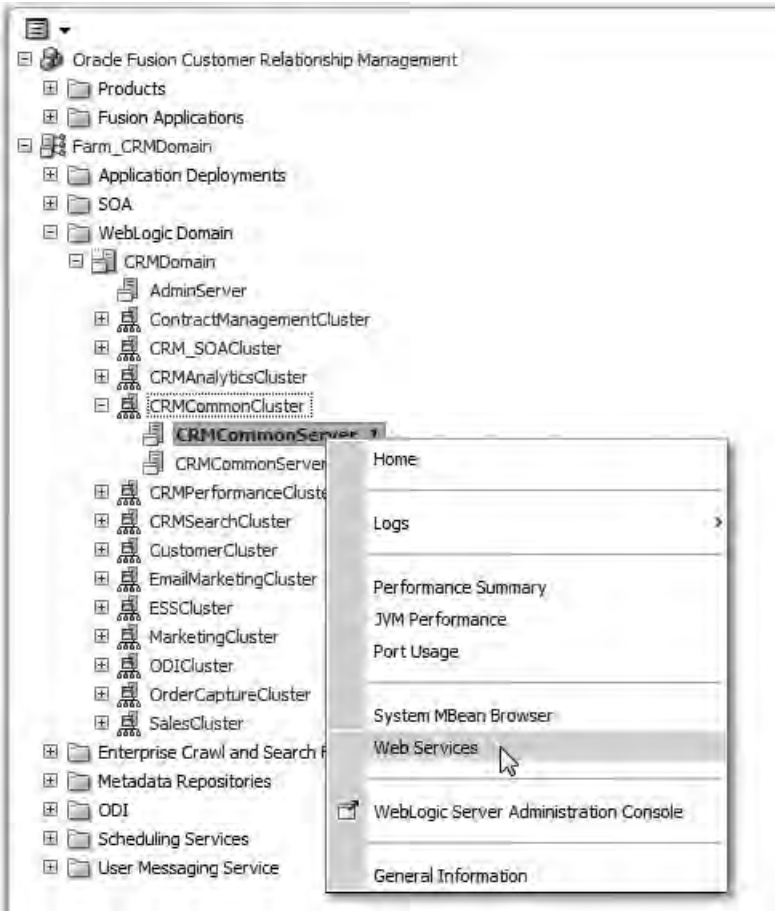
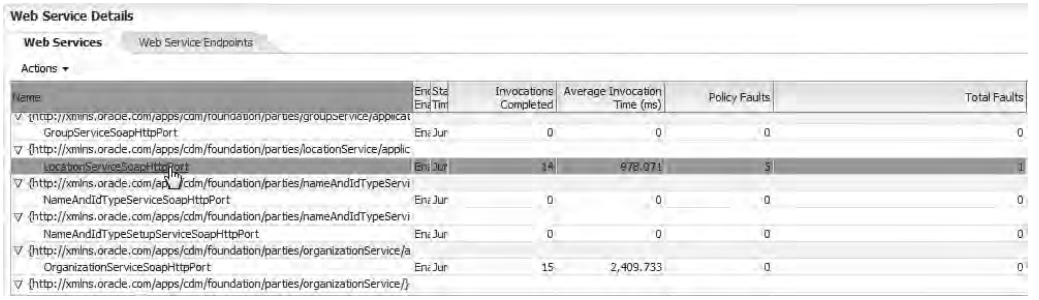


FIGURE 15-34. *Web services in Enterprise Manager*

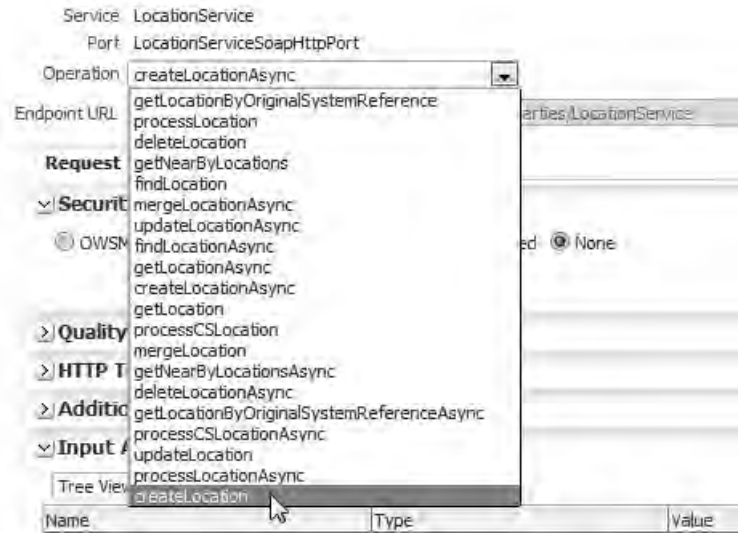
3. Locate the service you are interested in testing from the list and click on the service as shown in the following illustration. We will use location service as an example.



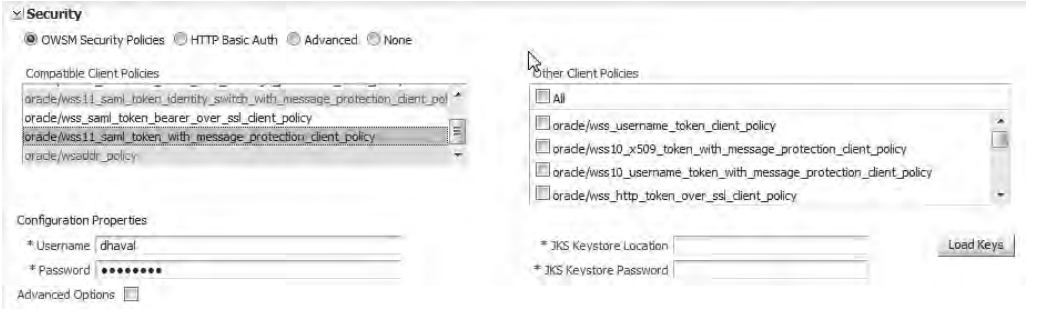
4. The service page shows details about the service, such as operations, the security policy used by the service, and other configuration details. You can click on the Web Services Test link at the top of the page as shown in Figure 15-35.

FIGURE 15-35. *Getting to the test service page in Enterprise Manager*

5. On the service test page, choose the Operation createLocation.



6. Under the security section, choose OWSM Security Policies and select the right policy for this service as shown in the following illustration. Provide a username and password for the user who has access to this service.



7. Under the input arguments section, you can see all the attributes available that you can input to this service in tree view. You can type in the values for

mandatory attributes per documentation. You can switch to XML view and enter the following input payload.

```
<soap:Envelope xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/">
    <soap:Body>
        <ns1:createLocation xmlns:ns1="http://xmlns.oracle.com/apps/cdm/foundation/
parties/
locationService/applicationModule/types/">
            <ns1:location xmlns:ns2="http://xmlns.oracle.com/apps/cdm/foundation/
parties/
locationService/">
                <ns2:Country>IN</ns2:Country>
                <ns2:Address1>Address1</ns2:Address1>
                <ns2:City>City</ns2:City>
                <ns2:PostalCode>PostalCode</ns2:PostalCode>
                <ns2:CreatedByModule>AMS</ns2:CreatedByModule>
            </ns1:location>
        </ns1:createLocation>
    </soap:Body>
</soap:Envelope>
```

8. Click the Test Web Service button from the top or bottom-right corner of the page. This will invoke the service with the given username and provide the result in the Response tab as shown here.

```
<env:Envelope xmlns:env="http://schemas.xmlsoap.org/soap/
envelope/" xmlns:wsa="http://www.w3.org/2005/08/addressing">
      <env:Header>
      <env:Body xmlns:wsu="http://docs.oasis-open.org/wss/2004/01/
oasis-200401-wss-wssecurity-utility-1.0.xsd" wsu:Id="Body-2Tplx6O0WK501GGh6ijVKw22">
            <ns0:createLocationResponse xmlns:ns0="http://xmlns.oracle.com/apps/cdm/
foundation/parties/locationService/applicationModule/types/">
                  <ns2:result xmlns:ns0="http://xmlns.oracle.com/adf/svc/types/"
xmlns:ns1="http://xmlns.oracle.com/apps/cdm/foundation/parties/
locationService/" xmlns:ns2="http://xmlns.oracle.com/apps/cdm/foundation/parties/
locationService/applicationModule/types/" xmlns:ns3="http://xmlns.oracle.com/apps/
cdm/foundation/parties/
partyService/" xmlns:ns4="http://xmlns.oracle.com/apps/cdm/foundation/parties/
flex/location/" xmlns:tns="http://xmlns.oracle.com/adf/svc/errors/"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
 xsi:type="ns1:LocationResult">
                        <ns1:Value>
                              <ns1:LocationId>300100020720793</ns1:LocationId>
                              <ns1:LastUpdateDate>2013-06-08T00:37:48.183-07:00
</ns1:LastUpdateDate>
                              <ns1:LastUpdatedBy>DHAVAL</ns1:LastUpdatedBy>
                              <ns1:CreationDate>2013-06-08T00:37:48.009-07:00
</ns1:CreationDate>
```

```
                              <ns1:CreatedBy>DHAVAL</ns1:CreatedBy>
            <ns1:LastUpdateLogin>DEA07C1FD59B2F9EE0431120F00A6EA0
</ns1:LastUpdateLogin>
                              <ns1:RequestId xsi:nil="true"/>
                              <ns1:OrigSystem xsi:nil="true"/>
<ns1:OrigSystemReference>300100020720793
</ns1:OrigSystemReference>
                              <ns1:Country>IN</ns1:Country>
                              <ns1:Address1>Address1</ns1:Address1>
                              <ns1:Address2 xsi:nil="true"/>
                              <ns1:Address3 xsi:nil="true"/>
                              <ns1:Address4 xsi:nil="true"/>
                              <ns1:City>City</ns1:City>
                              <ns1:PostalCode>PostalCode</ns1:PostalCode>
                              <ns1:State xsi:nil="true"/>
                              <ns1:Province xsi:nil="true"/>
                              <ns1:County xsi:nil="true"/>
                              <ns1:AddressStyle xsi:nil="true"/>
                              <ns1:ValidatedFlag>false</ns1:ValidatedFlag>
                              <ns1:AddressLinesPhonetic xsi:nil="true"/>
                              <ns1:PostalPlus4Code xsi:nil="true"/>
                              <ns1:Position xsi:nil="true"/>
                              <ns1:LocationDirections xsi:nil="true"/>
                              <ns1:AddressEffectiveDate xsi:nil="true"/>
                              <ns1:AddressExpirationDate xsi:nil="true"/>
                              <ns1:ClliCode xsi:nil="true"/>
                              <ns1:Language xsi:nil="true"/>
                              <ns1:ShortDescription xsi:nil="true"/>
                              <ns1:Description xsi:nil="true"/>
                              <ns1:SalesTaxGeocode xsi:nil="true"/>
                  <ns1:SalesTaxInsideCityLimits>1</ns1:SalesTaxInsideCityLimits>
                              <ns1:FaLocationId xsi:nil="true"/>
                              <ns1:ObjectVersionNumber>1</ns1:ObjectVersionNumber>
                              <ns1:CreatedByModule>AMS</ns1:CreatedByModule>
                              <ns1:ValidationStatusCode xsi:nil="true"/>
                              <ns1:DateValidated xsi:nil="true"/>
                              <ns1:DoNotValidateFlag xsi:nil="true"/>
                              <ns1:Comments xsi:nil="true"/>
                              <ns1:HouseType xsi:nil="true"/>
                              <ns1:EffectiveDate>2013-06-08</ns1:EffectiveDate>
                              <ns1:AddrElementAttribute1 xsi:nil="true"/>
                              <ns1:AddrElementAttribute2 xsi:nil="true"/>
                              <ns1:AddrElementAttribute3 xsi:nil="true"/>
                              <ns1:AddrElementAttribute4 xsi:nil="true"/>
                              <ns1:AddrElementAttribute5 xsi:nil="true"/>
```

```
                        <ns1:Building xsi:nil="true"/>
                        <ns1:FloorNumber xsi:nil="true"/>
                        <ns1:StatusFlag>true</ns1:StatusFlag>
                        <ns1:InternalFlag>false</ns1:InternalFlag>
                        <ns1:TimezoneCode xsi:nil="true"/>
                    </ns1:Value>
                </ns2:result>
            </ns0:createLocationResponse>
        </env:Body>
    </env:Envelope>
```

How to Change the User in SOA Composite to Call Services

In our example in the preceding section, we did not specify any user when we called the service from the BPEL process in the SOA composite. By default, SOA composites do identity propagation; that is, the composite will invoke the service with the exact same user identity that resulted in the instantiation of the composite. In our example, the SOA composite is instantiated by a Create Location event, which is raised when a user created data in the application UI. The service from the SOA composite is called with the exact same user. In our example, we are calling the service from the same Fusion Applications installation, so the integration works seamlessly. You will be faced with situations many times where you will need to switch the identity of the user before making the service call from the SOA composite because the systems you are integrating are not the same. There are two ways to do this identity switch in the SOA composite.

Using a Hard-Coded Username and Password

If you know the username and password for a user who has access to call a given service from a given application, you can simply pass that fixed username and password for calling the service. You can specify the values as properties in the SOA composite service reference as shown here.

```
    <binding.ws port="http://xmlns.oracle.com/apps/cdm/foundation/parties/
locationService/applicationModule/
#wsdl.endpoint(LocationService/LocationServiceSoapHttpPort)"
    location="oramds:/apps/oracle/apps/cdm/foundation/parties/
locationService/LocationService.wsdl">
      <wsp:PolicyReference URI="oracle/wss11_saml_token_with_
message_protection_client_policy"
                           orawsp:category="security" orawsp:status="enabled"/>
      <property name="basicHeaders">credentials</property>
      <property name="basicUsername">dhaval</property>
      <property name="basicPassword">Welcome1</property>
    </binding.ws>
```

Using Keystore Configuration

The hard-coded way to pass the username and password may not sound secure, and you may want to use a more robust mechanism to do the user switch. Please read *Oracle Fusion Middleware Security and Administrator's Guide for Web Services* to understand how to generate the csf-key to use with a Web service. This is a standard way to generate secured tokens that can be used to call a service, and it will work only for a given server configuration. Once you have the csf-key generated from your security administrator, you can use it in your SOA composite as shown here.

```
    <binding.ws port="http://xmlns.oracle.com/apps/cdm/foundation/parties/
locationService/applicationModule/
#wsdl.endpoint(LocationService/LocationServiceSoapHttpPort)"
    location="oramds:/apps/oracle/apps/cdm/foundation/parties/
locationService/LocationService.wsdl">
      <wsp:PolicyReference URI="oracle/wss11_saml_token_with_message_protection_
client_policy"
                           orawsp:category="security" orawsp:status="enabled"/>
      <property name="csf-key" many="false">your-csf-key</property>
    </binding.ws>
```

Summary

In this chapter, we discussed the different integration options available for Fusion Applications for inbound and outbound interaction. We discussed how to discover the assets using OER. We discussed various service options available in Fusion Applications and how the service-oriented architecture can be used to do integrations using business events and Web services. We also walked through a complete end-to-end example of integrating two applications for data sync. We examined how to deploy the SOA composite and how to test and examine the flow using Enterprise Manager. Finally, we talked about how to switch users for calling services from the SOA composite and how to test Fusion Applications Web services.