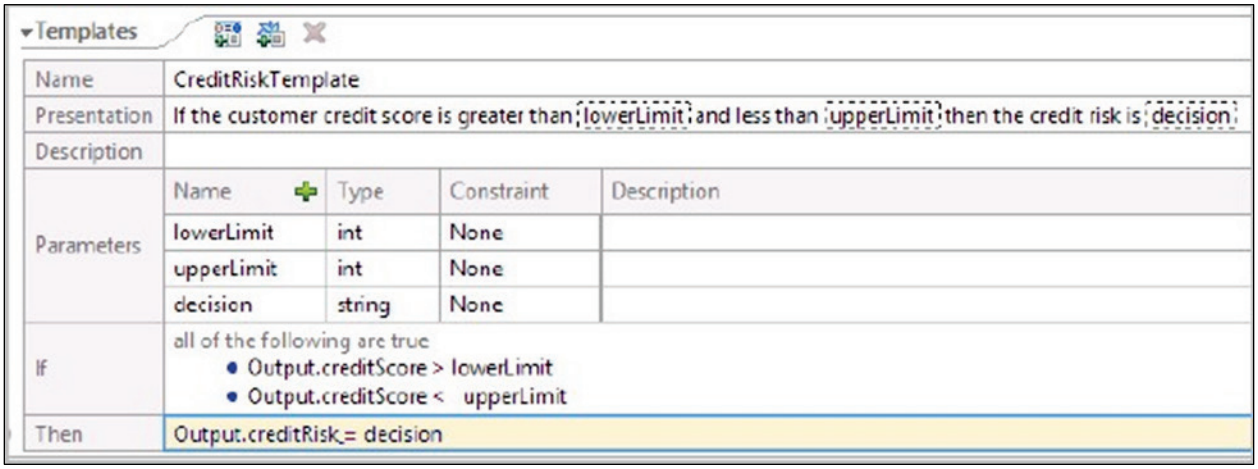
34. Save your changes.

The completed template resembles the following figure:



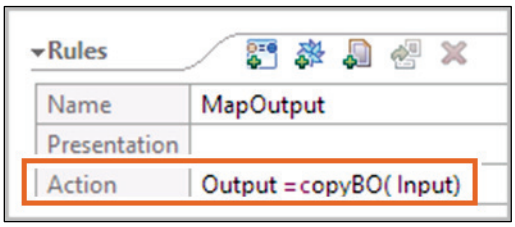
7. Create an initialization action rule that is named `MapOutput` that copies the data from the Input business object and assigns it to the Output business object.

The rule logic is implemented by using the following expression: `Output = copyBO(Input)`

1. In the **Rules** section of the editor, click the **Add Action Rule** icon.

2. Change the **Name** from `Rule1` to: `MapOutput`
3. Select the **Action** link and select **Output : CustomerApplication** from the list.
4. Select the equal sign (**=**) operator.
5. Select **Copy BO**.
6. Select **Input : CustomerApplication**.

The completed “Action” section of the rule resembles the following figure:

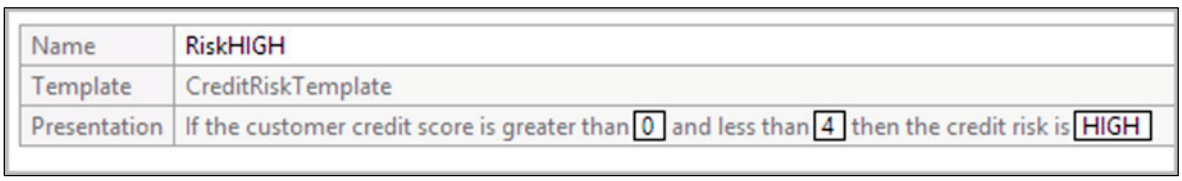


8. Implement the logic of the rule set by creating the following business rules. Use the CreditRiskTemplate you created previously to generate the rules.

   - If the **creditScore** value is less than 4, then the **creditRisk** is `HIGH`.
   - If the **creditScore** value is in the range of 4 – 7, then the **creditRisk** is `MED`.
   - If the **creditScore** value is in the range of 8 – 11, then the **creditRisk** is `LOW`.
   - Otherwise, set the **creditRisk** to `HIGH`.

1. In the **Rules** section of the editor, click the **Add Template Rule** icon.

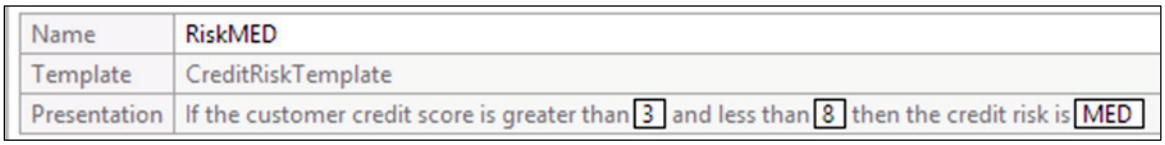
2. Select **CreditRiskTemplate** from the list.
3. In the **Name** field, replace `Rule1` with: `RiskHIGH`
4. In the **Presentation** field, click the **Enter Value** link in the first text box.
5. Type `0` in the box.
6. In the **Presentation** field, click the **Enter Value** link in the second text box.
7. Type `4` in the box.
8. In the **Presentation** field, click the **Enter Value** link in the third text box.
9. Type `HIGH` in the box. The completed rule resembles the following figure:



10. Click the **Add Template Rule** icon to add another rule to the rule set.

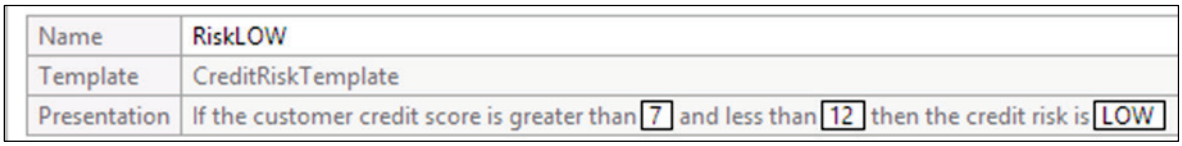
11. Select **CreditRiskTemplate** from the list.
12. In the **Name** field, replace `Rule1` with: `RiskMED`
13. In the **Presentation** field, click the **Enter Value** link in the first text box.
14. Type `3` in the box.
15. In the **Presentation** field, click the **Enter Value** link in the second text box.
16. Type `8` in the box.
17. In the **Presentation** field, click the **Enter Value** link in the third text box.

18. Type `MED` in the box.
    The completed rule resembles the following figure:

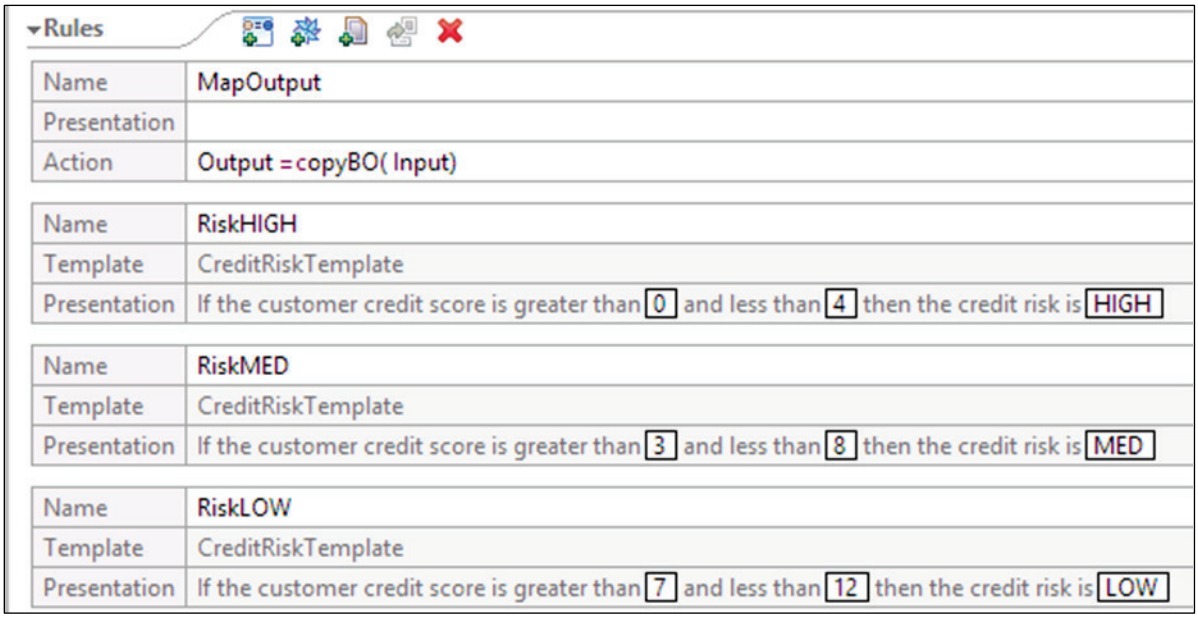


19. In the **Rules** section of the editor, click the **Add Template Rule** icon.

20. Select **CreditRiskTemplate** from the list.
21. In the **Name** field, replace `Rule1` with: `RiskLOW`
22. In the **Presentation** field, click the **Enter Value** link in the first text box.
23. Type `7` in the box.
24. In the **Presentation** field, click the **Enter Value** link in the second text box.
25. Type `12` in the box.
26. In the **Presentation** field, click the **Enter Value** link in the third text box.
27. Type `LOW` in the box.
    The completed rule resembles the following figure:



28. Save your changes.
    The completed rule set resembles the following figure:

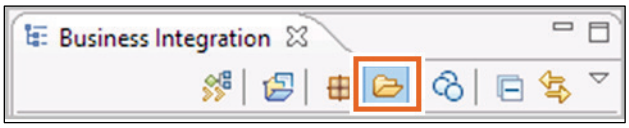


29. Close the rule set editor. Continue to ignore any errors in the **Problems** view.

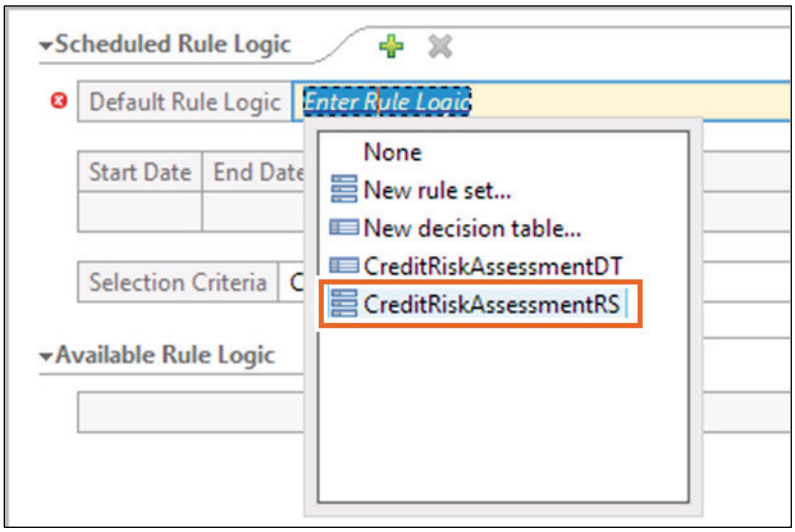
9. Add CreditRiskAssessmentRS to the CreditRiskAssessment rule group as the Default Rule Logic, and add CreditRiskAssessmentDT to the Available Rule Logic.

1. In the Business Integration view, expand FoundationServices > Integration Logic > Rule Groups > businessrules.
2. Double-click **CreditRiskAssessment** to open the rule group editor.

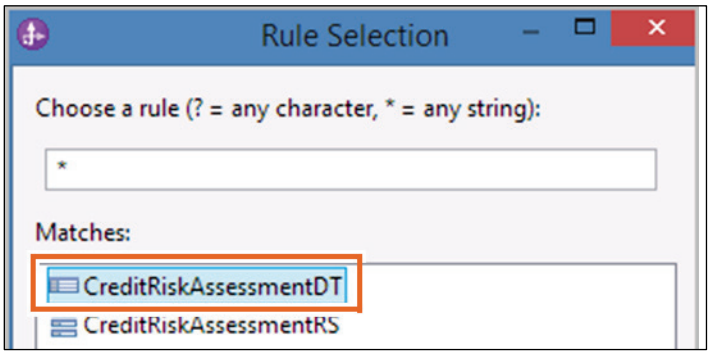
   Note: If the businessrules folder is not displayed in the view, click the "Show/Hide Folders" icon.



3. Click the **Enter Rule Logic** link in the **Default Rule Logic section**.
4. Select **CreditRiskAssessmentRS** from the list.



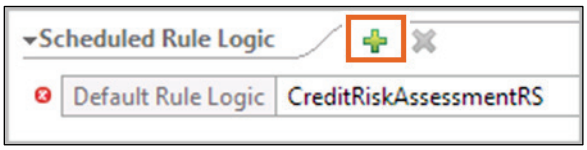
5. In the **Available Rule Logic** section of the editor, click the plus sign (**+**) icon.
6. Select **CreditRiskAssessmentDT** in the Rule Selection dialog box.



7. Click **OK** to add the decision table to the Available Rule Logic section.

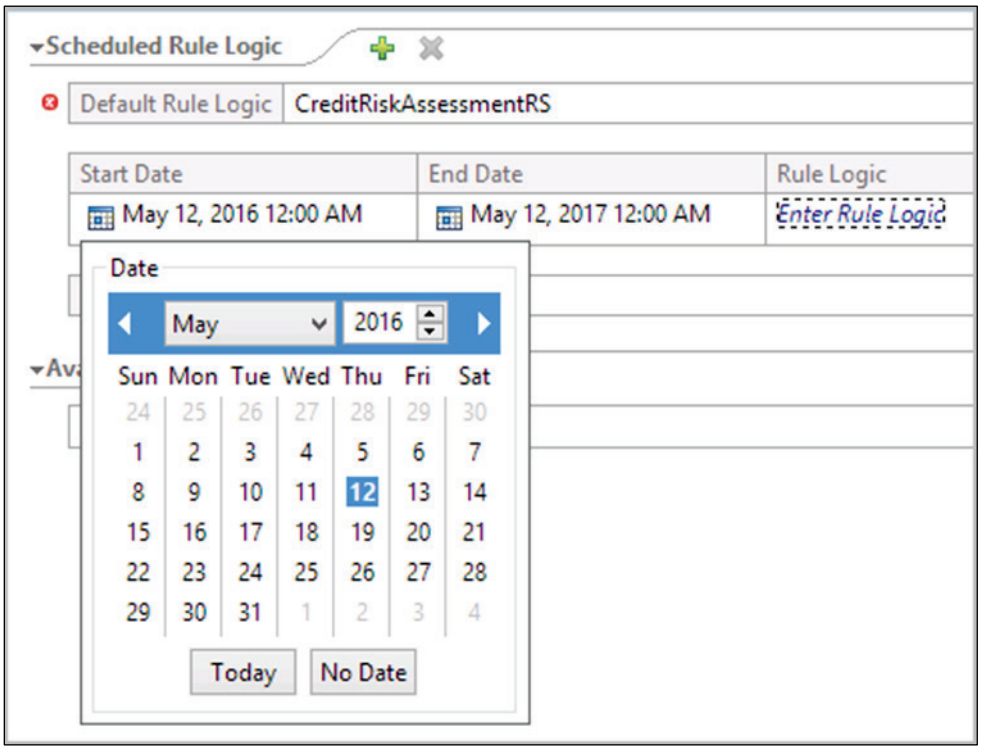
10. Complete the Start Date, End Date, and Rule Logic entries in the selection table so that the CreditRiskAssessmentRS rule set is invoked when the Current date is between 12 AM today and 11:59 PM one year from now.

1. Click the **Add Date Selection Entry** icon (the green plus icon) to set the **Start Date** and **End Date** for the target rule logic.



2. Click the **Start Date** calendar icon and if the date is not set automatically, select the current date.

   The dates that you choose are not going to be the same dates that are shown here.

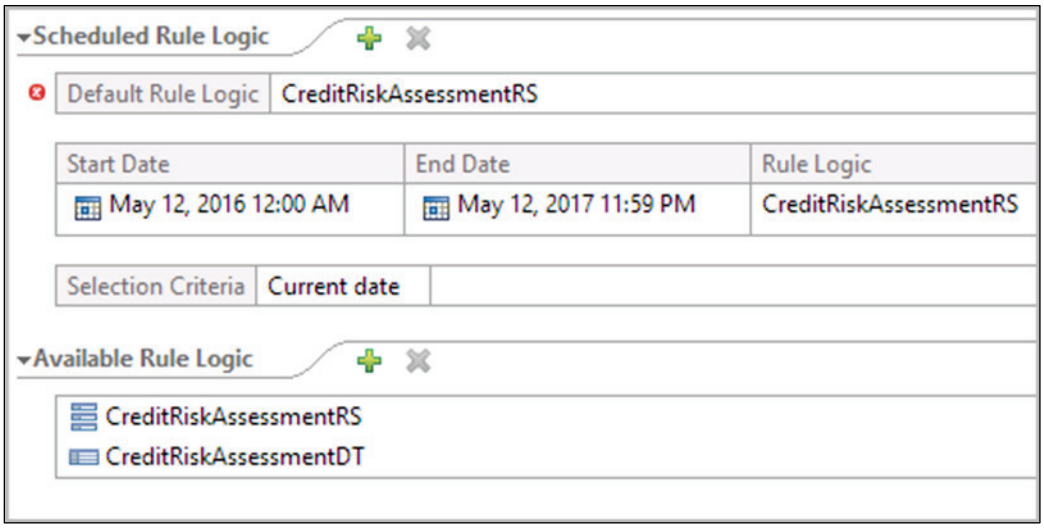


   The time is set to **12 AM** automatically.

3. Click the **End Date** calendar icon and select a date one year from today.
4. For the End Date value, place your cursor inside **12:00 AM** and manually change the time to: `11:59 PM`
5. Click the **Enter Rule Logic** link in the **Rule Logic** column of the table.

6. Select **CreditRiskAssessmentRS** from the list.

   The completed Scheduled Rule Logic section of the rule group resembles the following figure:



   Information: For this exercise, the provided examples use the date range that starts May 12, 2016 and ends May 12, 2017. The dates that you choose are not going to be the same dates.

7. Save your changes and close the rule group editor. Verify that the **Problems** view shows no errors. You can ignore any warnings.

Part 2. Incorporate a rule group component into an assembly diagram.

In this portion of the exercise, you add the CreditRiskAssessment rule group to the FoundationServices assembly diagram, and you generate an export component for the rule group with an SCA binding. This action allows other IBM Process Server SCA applications to call FoundationServices in the most efficient manner possible.

You then create an import component on the FoundationModule assembly diagram and wire it to the CreditRiskAssessmentPartner reference of the AccountVerification business process. When the Credit Risk Assessment invoke activity is reached, the process uses the import component that is wired to the CreditRiskAssessmentPartner reference to call the export component in FoundationServices.

To assemble the application and test the business rules:

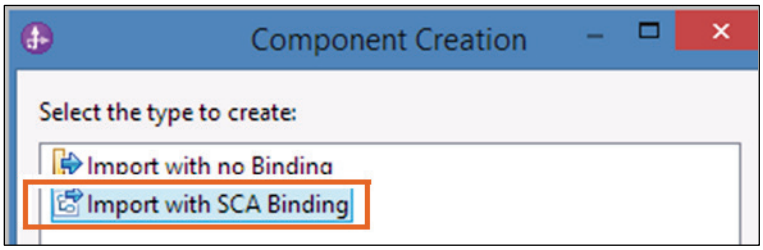
1. Add the CreditRiskAssessment rule group component to the FoundationServices assembly diagram.

1. If the assembly diagram is not already open, in the Business Integration view, expand **FoundationServices** and double-click **Assembly Diagram**.

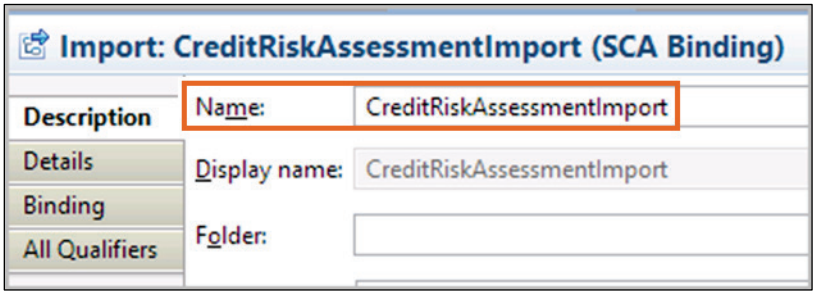
2. In the Business Integration view, expand **FoundationServices > Integration Logic > Rule Groups > businessrules**.
3. Drag the **CreditRiskAssessment** rule group onto the assembly diagram.
   You will generate an export component for the CreditRiskAssessment rule group with an SCA binding.
4. On the FoundationServices assembly diagram, right-click **CreditRiskAssessment** and click **Generate Export > SCA binding** from the menu.
5. Accept the default export name: `CreditRiskAssessmentExport`
6. Save the assembly diagram.

2. Create an import component on the FoundationModule assembly diagram that is used to invoke the CreditRiskAssessmentExport in FoundationServices.

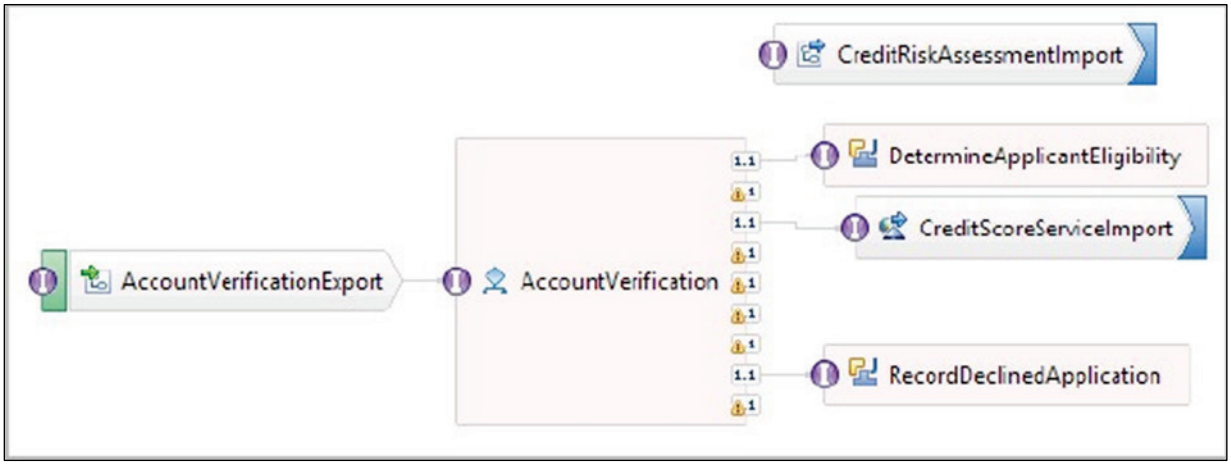
1. In the Business Integration view, expand **FoundationModule** and double-click **Assembly Diagram.**
2. In the Business Integration view, expand **FoundationServices > Assembly Diagram**.
3. Drag **CreditRiskAssessmentExport** onto the **FoundationModule** assembly diagram.
4. In the Component Creation dialog box, select **Import with SCA Binding**.



5. Click **OK**.
6. Select the newly created **Import1** and switch to the **Description** tab in the **Properties** view.
7. Change the **Name** from `Import1` to: `CreditRiskAssessmentImport`

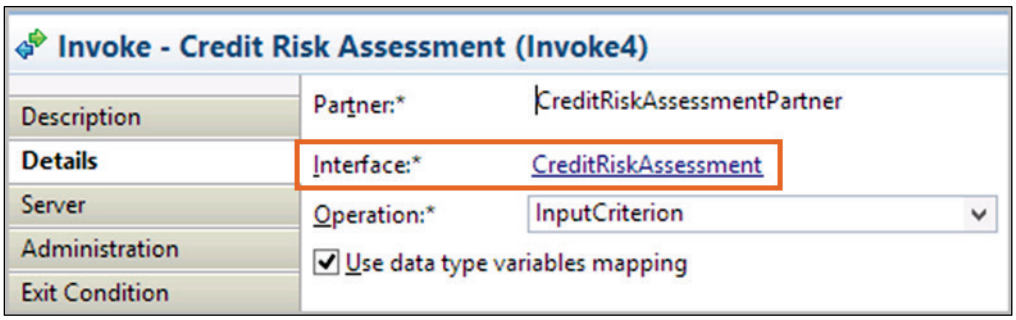


8. Save your changes to the assembly diagram. **AccountVerification** has three references that are wired to target services. The **CreditRiskAssessmentImport** component is going to be wired soon.



3. Examine the process activity that invokes the CreditRiskAssessment rule group.

1. Double-click the **AccountVerification** component in the assembly diagram to open the process implementation in the business process editor.
2. Select the **CreditRiskAssessment** invoke activity in the business process.
3. Switch to the **Details** tab in the **Properties** view. The activity invokes the **CreditRiskAssessment** interface of the **CreditRiskAssessment** rule group.

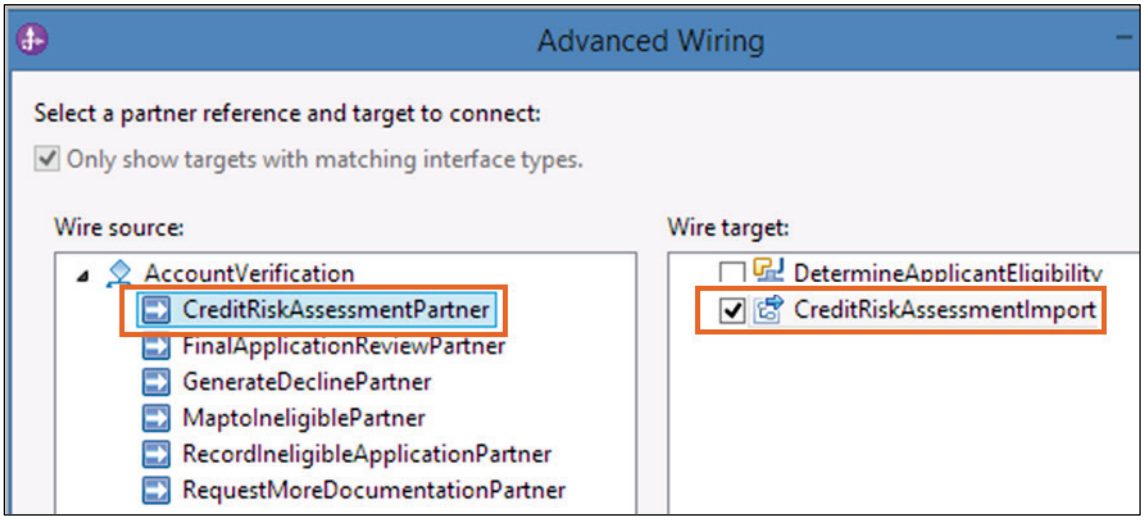


4. Close the business process editor and return to the **FoundationModule** assembly diagram.

4. Wire the CreditRiskAssessmentImport component to the CreditRiskAssessmentPartner reference on the AccountVerification process component.

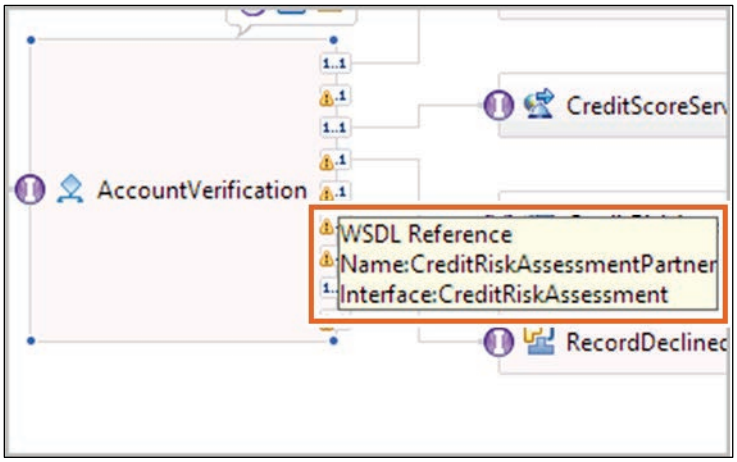
1. Right-click the **AccountVerification** process component and click **Wire (Advanced)** from the menu.
2. In the Advanced Wiring dialog box, in the “Wire source” window, select **CreditRiskAssessmentPartner.**

3. In the “Wire target” window, select **CreditRiskAssessmentImport**.

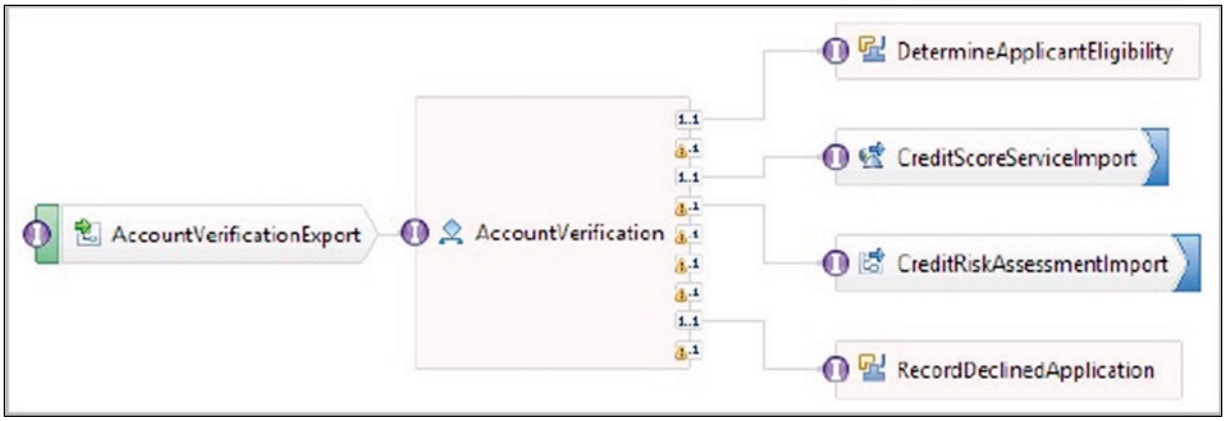


4. Click **OK**.
5. Save your changes.

   You can verify the wiring by hovering over the reference that is wired to **CreditRiskAssessmentImport**. The dialog box indicates that the component is wired to **CreditRiskAssessmentPartner**.



The assembly diagram resembles the following figure:



Part 3. Test a business rule group in the integrated test client.

After assembling the application, you test the CreditRiskAssessment business rules by invoking the CreditRiskAssessmentExport component.

To test the business rules:

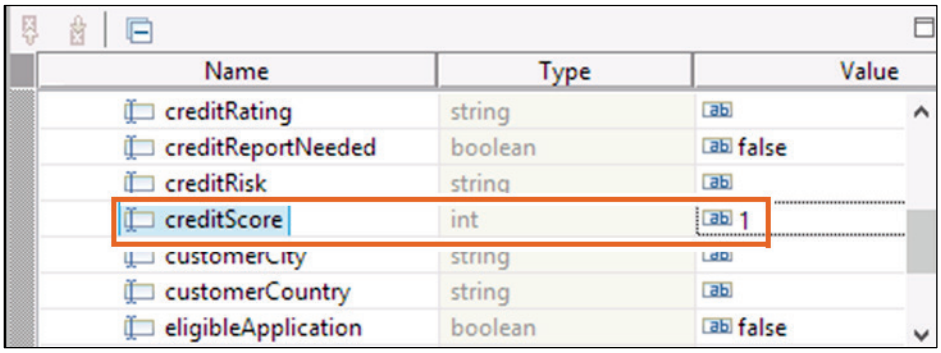
1. Start the server (if it is not already running) and deploy FoundationServicesApp.

1. Start the server, if not running, by double-clicking the desktop shortcut.
2. In the **Servers** view, right-click **IBM Process Server v8.6 at localhost** and click **Add and Remove** from the menu.
3. Select **FoundationServicesApp** in the **Available projects** window and click **Add**.
4. Click **Finish**.
5. Wait until the module is published and started. The process is complete when the message `Application started: FoundationServicesApp` is displayed in the **Server Logs** view.

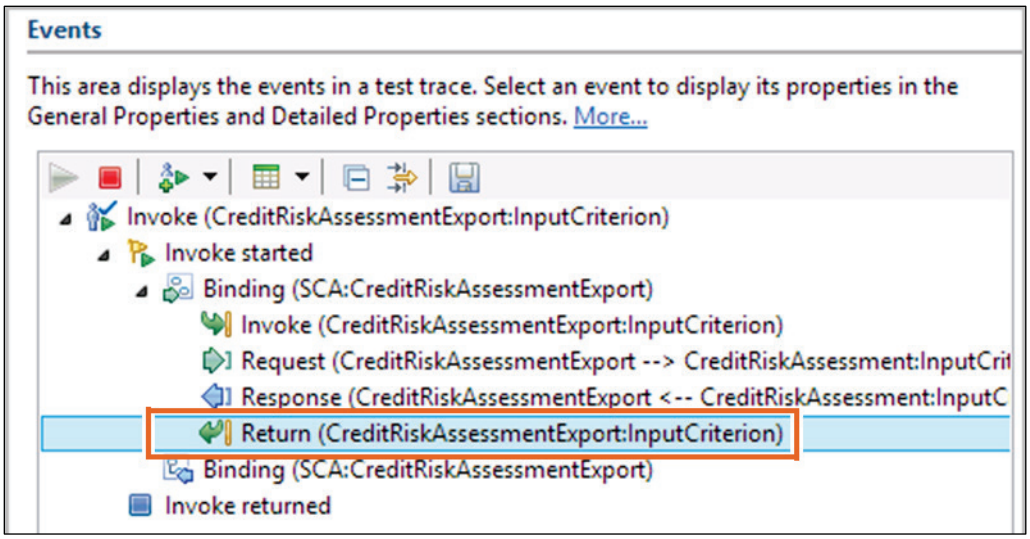
2. Use several different creditScore values to test the CreditRiskAssessmentExport in the FoundationServices module. The business rules return correct creditRisk values.

1. If the assembly diagram is not open, in the Business Integration view, expand **FoundationServices** and double-click **Assembly Diagram**.
2. In the FoundationServices assembly diagram, right-click **CreditRiskAssessmentExport** and click **Test Component**.
3. In the **Initial request parameters** section, change the **creditScore** value to 1.

   It is not necessary to populate any of the remaining fields.

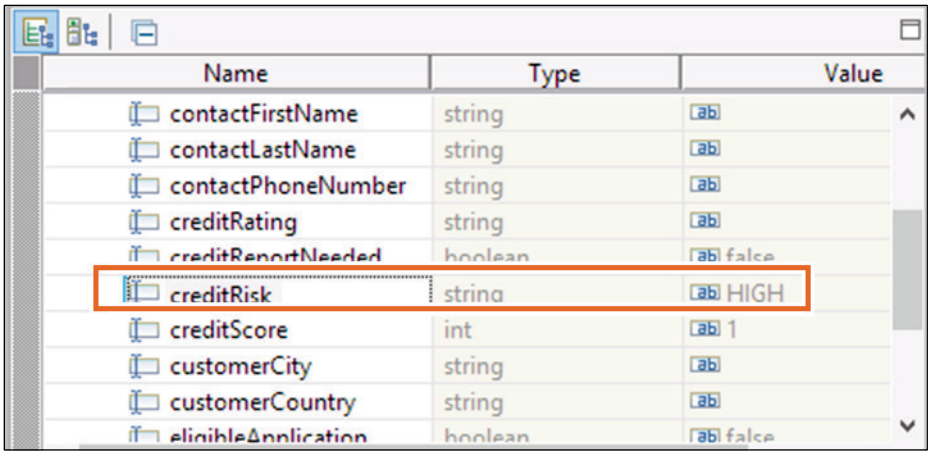


4. Click **Continue** to run the test.
5. When the **Select a Deployment Location** dialog box is displayed, select **IBM Process Server v8.6 at localhost** and click **Finish**.
6. When the User Login dialog box is displayed, accept the default values for the **User ID** and **Password** fields. Click **OK**.
7. If it is not already selected, select the **Return(CreditRiskAssessmentExport:InputCriterion)** event in the **Events** window.



8. Review the output value for creditRisk in the **Return parameters** section.

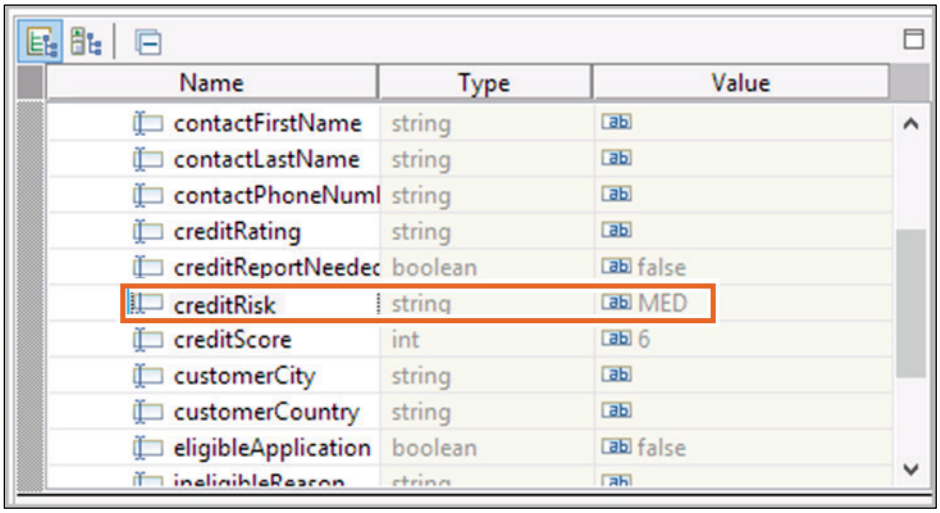
   According to the CreditRiskAssessmentRS definition, a credit score of less than 4 evaluates to a credit risk of `HIGH`.



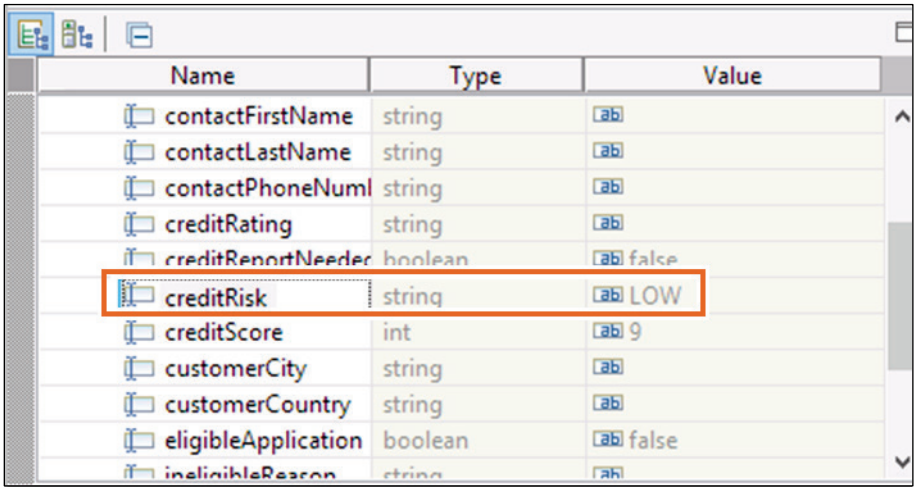
   You now rerun the test with credit score of 6, which evaluates to a creditRisk of `MED`.

9. Click the **Invoke** icon to create another test invocation thread.

10. In the **Initial request parameters** section, change the **creditScore** value to `6`.
11. Click **Continue** to run the test.
12. Review the **Return parameters** for the **Return(CreditRiskAssessmentExport:InputCriterion)** event and verify that the **creditRisk** is `MED`.



13. Repeat the steps to test a **creditScore** value of 9, which evaluates to a **creditRisk** of `LOW`.



14. When you finish, close the **FoundationServices_test** tab.
15. Click **No** when you are prompted to save the test trace.
16. Leave the server in the running state with **FoundationServicesApp** deployed.

Part 4. Use the Business Rule Manager web client to interact with business rules at run time.

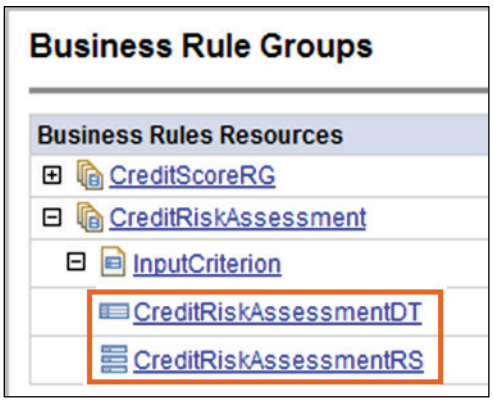
In this portion of the exercise, you use the Business Rules Manager web client to view the properties of your rule group and to do the necessary steps to create a rule from a template. Administrators and business users can use the Business Rules Manager web client to change rule parameters and add rules from templates, among other things. Here you learn how to make but not save the changes.

1. To view your rules in the Business Rules Manager:

1. Start the Business Rules Manager web client.
2. Start Firefox and enter `http://localhost:9080/br` in the URL field.
1. If the **This Connection is Untrusted** window is displayed, then expand **I Understand the Risks** and click **Add Exception**.
2. In the Add Security Exception window, click **Confirm Security Exception**.
3. In the login page for the Business Rules Manager, enter `admin` in the **User ID** field and `web1sphere` in the **Password** field, and click **Login**.

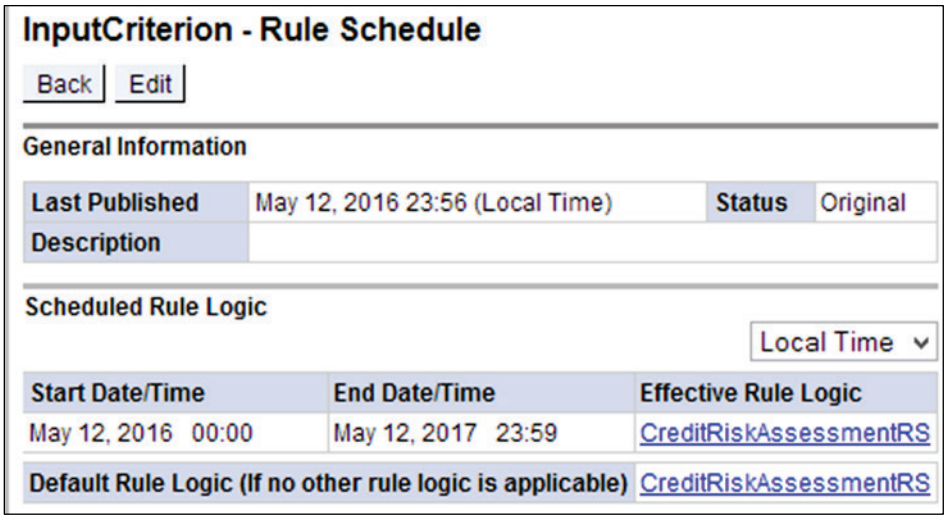
2. Examine the contents of the CreditRiskAssessment rule group in the Business Rules Manager web client.

1. Expand CreditRiskAssessment > InputCriterion to view the available rules.



2. Click the **InputCriterion** link and examine the content.

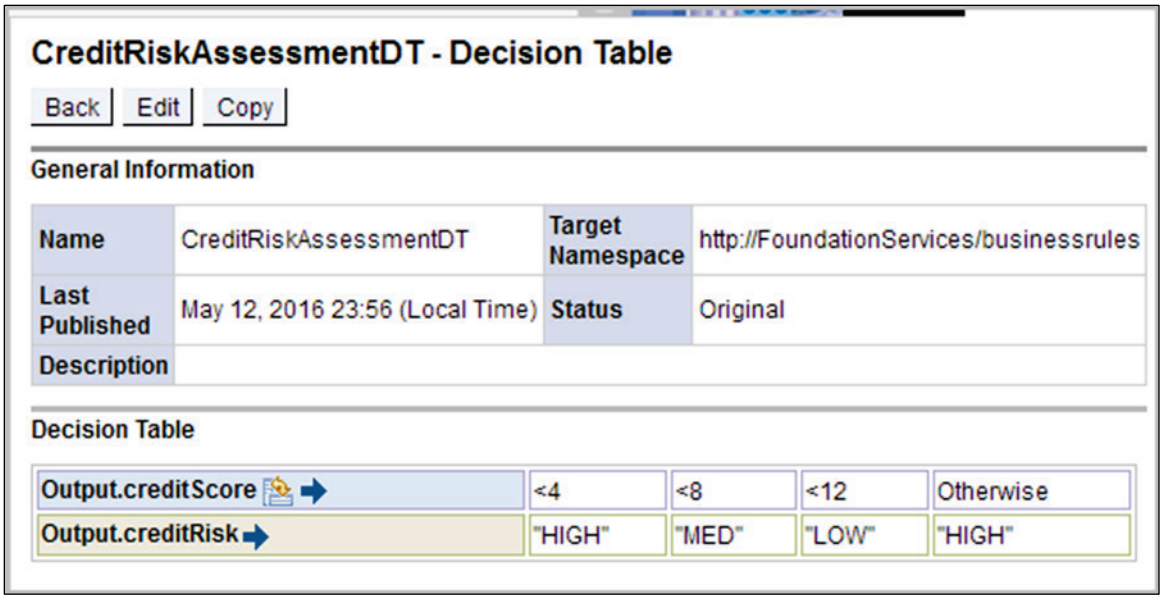
   This page shows the properties of the rule group (the Scheduled Rule Logic and Default Rule Logic).



3. Click **Back** at the top of the page to return to the previous page.

   Make sure that you do not click back for the browser because that throws an error. You must click **Back** on the page.

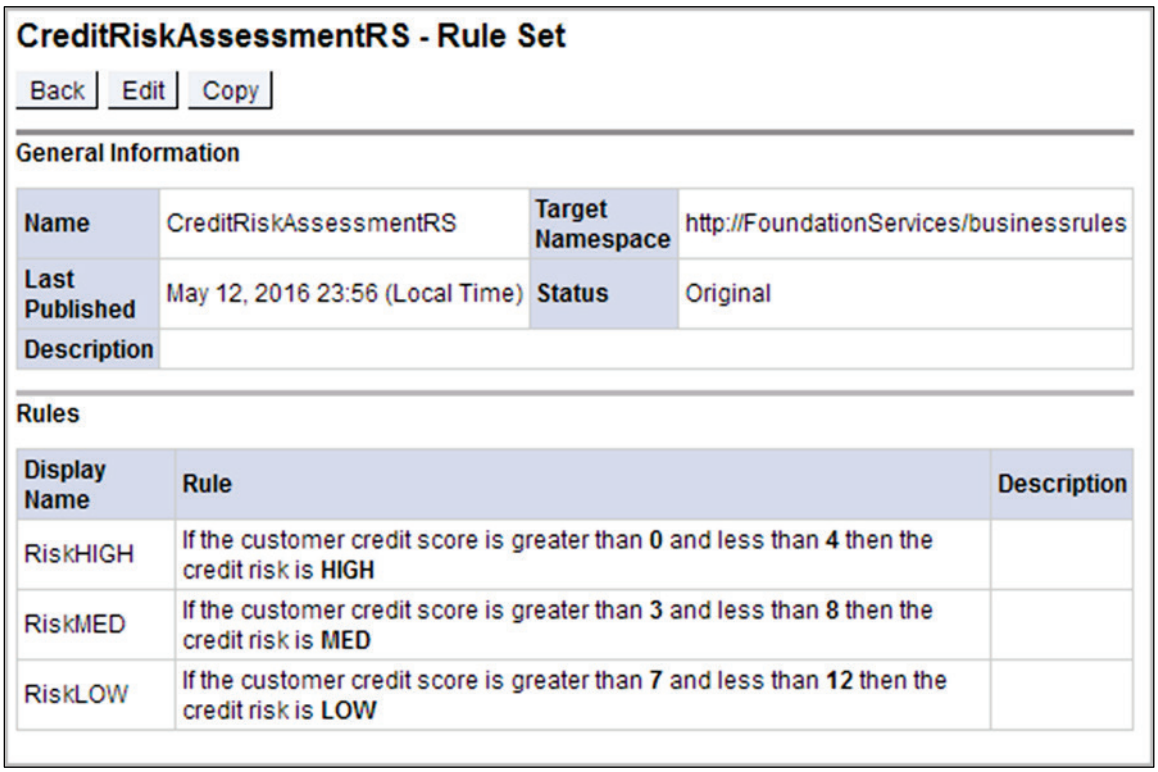
4. Click the **CreditRiskAssessmentDT** link and examine the content.



5. Click the **Back** button to return to the previous page.

   Make sure that you click **Back** on the page and not the back on the browser because that does not work for the business rules client.

6. Click the **CreditRiskAssessmentRS** link and examine the content.



CreditRiskAssessmentRS - Rule Set

Back | Edit | Copy

General Information

Name	CreditRiskAssessmentRS	**Target Namespace**	http://FoundationServices/businessrules
Last Published	May 12, 2016 23:56 (Local Time)	**Status**	Original
Description			

Rules

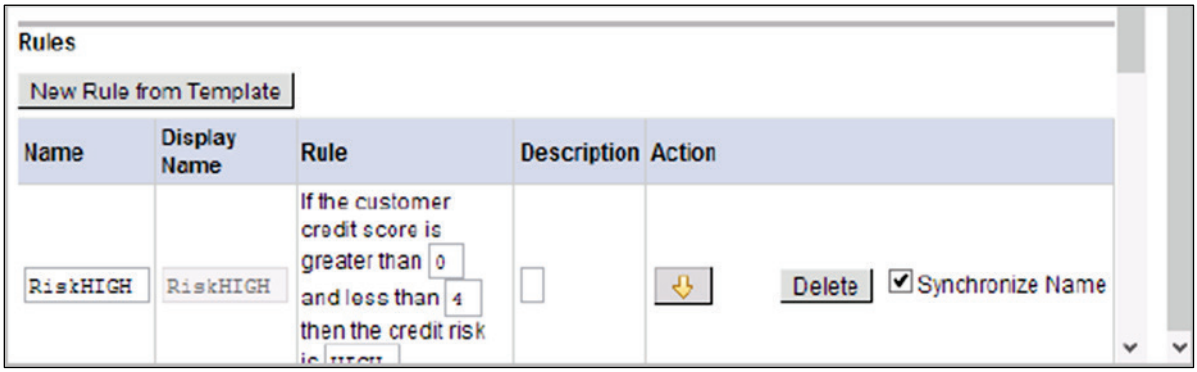
Display Name	Rule	Description
RiskHIGH	If the customer credit score is greater than **0** and less than **4** then the credit risk is **HIGH**	
RiskMED	If the customer credit score is greater than **3** and less than **8** then the credit risk is **MED**	
RiskLOW	If the customer credit score is greater than **7** and less than **12** then the credit risk is **LOW**	

7. Click **Edit**. I

   In edit mode, you can change descriptions, change individual rule parameters, change rule names, delete rules, or change the execution order by moving the rules up or down the list.

8. Click **New Rule from Template** to create a rule.

   In the Business Rules Manager, you can create rules only from templates. If the developer does not implement rule templates in the application, new rules cannot be added and rule parameters cannot be changed at run time.

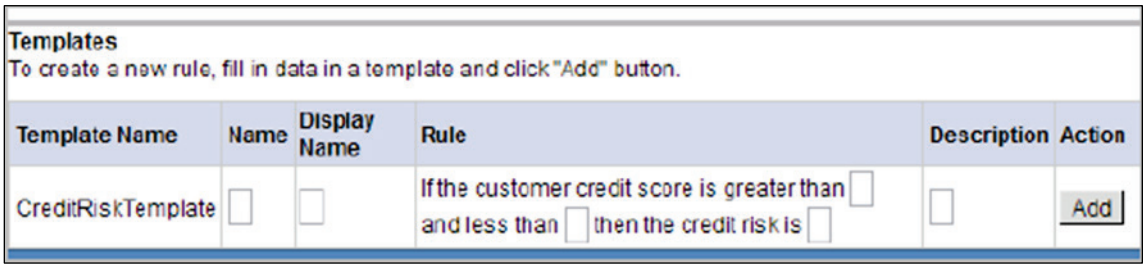


Rules

New Rule from Template

Name	Display Name	Rule	Description	Action
RiskHIGH	RiskHIGH	If the customer credit score is greater than 0 and less than 4 then the credit risk is HIGH		Delete Synchronize Name

9. You can create a rule by entering a **Name**, **Display Name**, **Rule parameters**, and a **Description** value.



10. Click **Close Template List** and click **Cancel**.

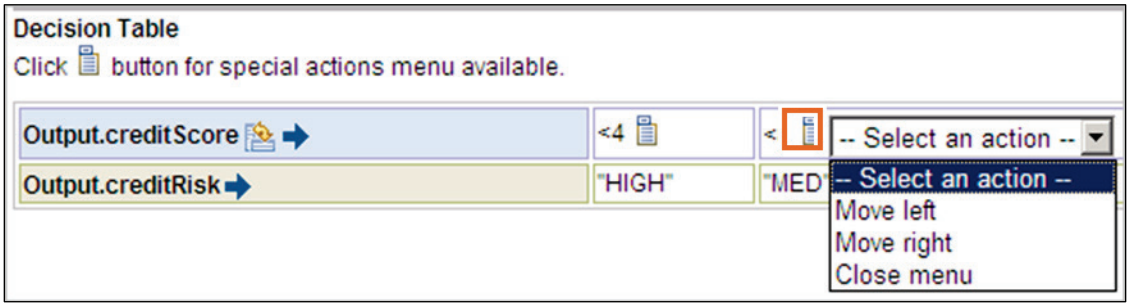
    This action prevents the rule from taking effect in the runtime environment. To apply the new rule immediately, you save the rule and publish it to the runtime environment.

    Changes that you make in the Business Rules Manager are not synchronized with IBM Integration Designer. To bring runtime changes into a project, you must use the administrative console to export the rules from the runtime environment.

    The following screen capture depicts the appropriate view in the administrative console. You are not required to do this task.

11. Click **Back** to return to the previous page.
12. Click the **CreditRiskAssessmentDT** link.
13. Click **Edit**.

14. Click one of the **Open the actions menu** icons to view the choices available.



15. Click **Cancel** to return to the previous page.
    Feel free to examine the application.
16. Click **Logout** to exit the Business Rules Manager web browser and close the browser.

3. Remove the projects and (optionally) stop the server.

1. In the Servers view, right-click IBM Process Server v8.6 at localhost and click Add and Remove.
2. Click **Remove All** and click **Finish**.
3. (Optional) Stop the server.
4. Close IBM Integration Designer.

Results:
In this exercise, you created rule sets and decision tables that contain business rules.

IBM Training

References

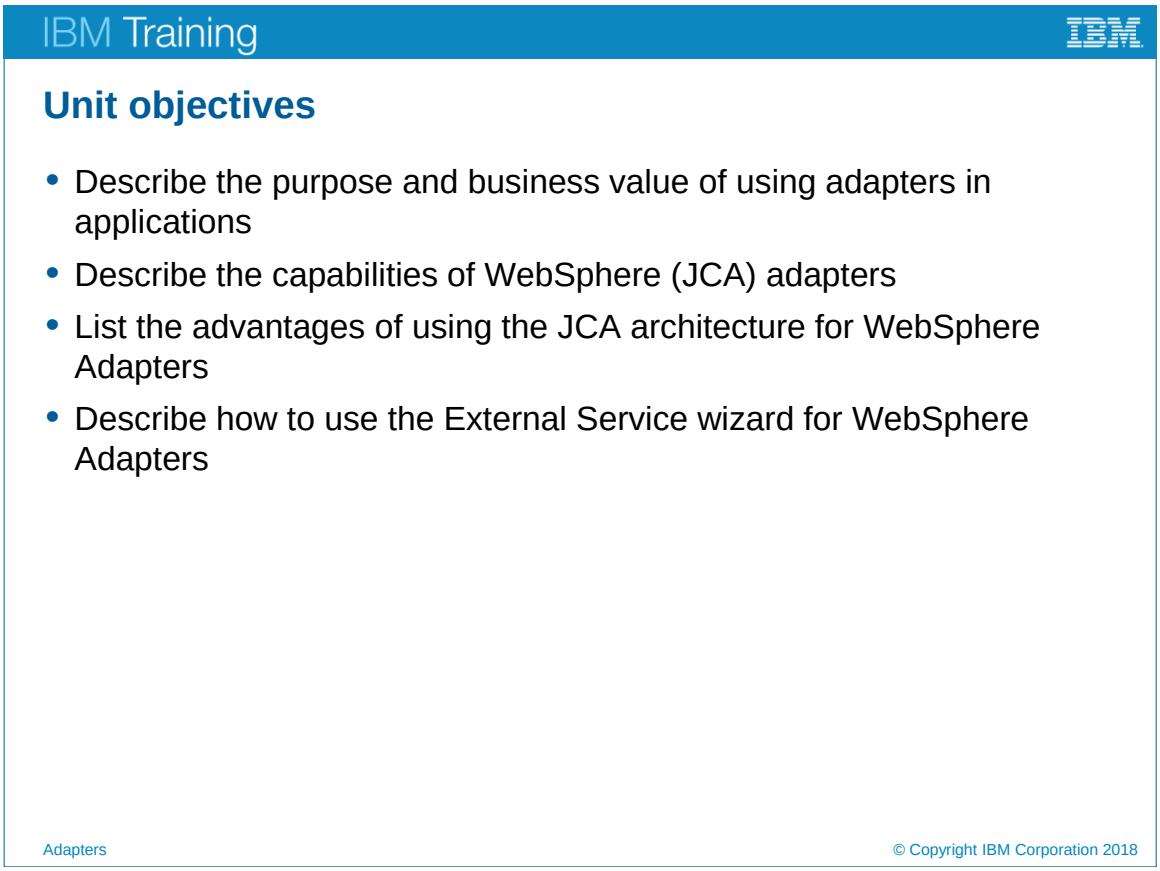
- Business Rules Community
  - http://www.brcommunity.com
- Business Rules Group
  - http://www.businessrulesgroup.org
- SOA programming model for implementing web services, part 9: Integrating rules with SOA
  - http://www.ibm.com/developerworks/webservices/library/ws-soa-progmodel9/index.html
- Creating and deploying business rules
  - http://www.ibm.com/developerworks/websphere/library/tutorials/0610_kolban/0610_kolban.html

References

IBM Training

Adapters

IBM Business Process Manager V8.6



Unit objectives

In this unit, you learn about WebSphere (JCA) adapters and how they are used to integrate solutions with other applications.

IBM Training

Topics

- Introduction to adapters
- Using WebSphere (JCA) Adapters

Topics

IBM Training

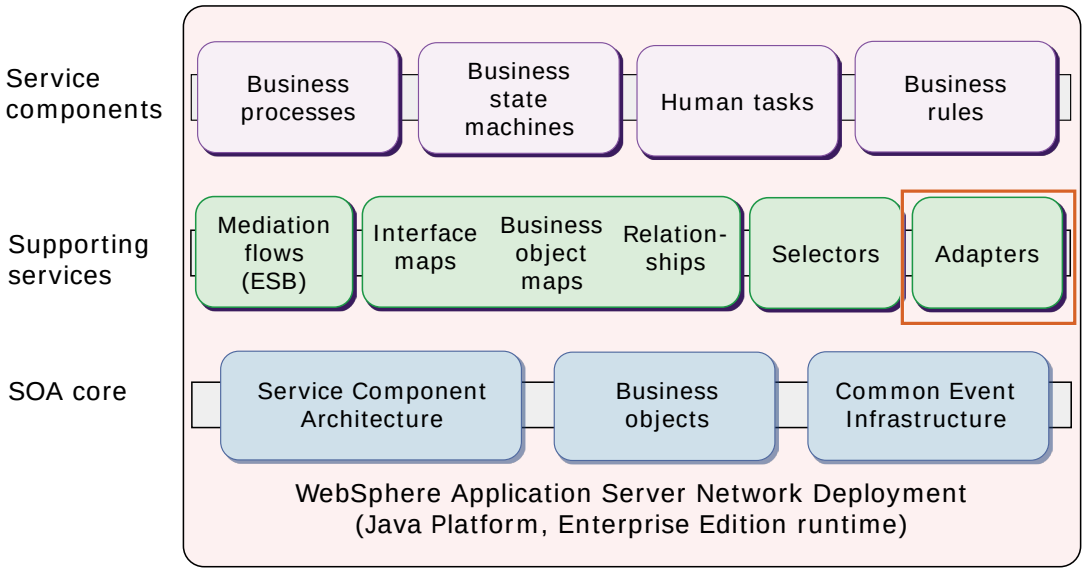
Introduction to adapters

Introduction to adapters

IBM Training

Adapters are supporting services

- Adapters are part of the supporting services layer



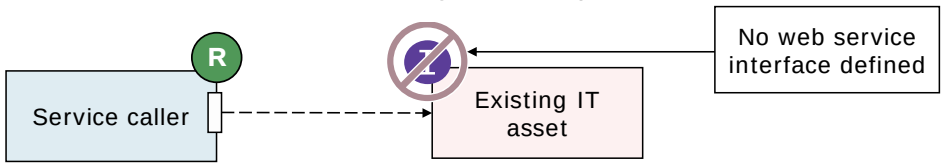
Adapters

Adapters are supporting services

IBM Training

Introduction to adapters

- Adapters provide integration with enterprise information systems (EIS) without service interfaces
- An EIS provides the information infrastructure for an enterprise by providing a set of services to clients:
  - Enterprise resource planning (ERP)
  - Customer relationship management (CRM)
  - Human resource systems (HR)
- Adapters use existing IT assets in your SOA without significant programming, which simplifies integration
  - Adapters are presented as SCA components
  - EIS is invoked as a service through the adapter



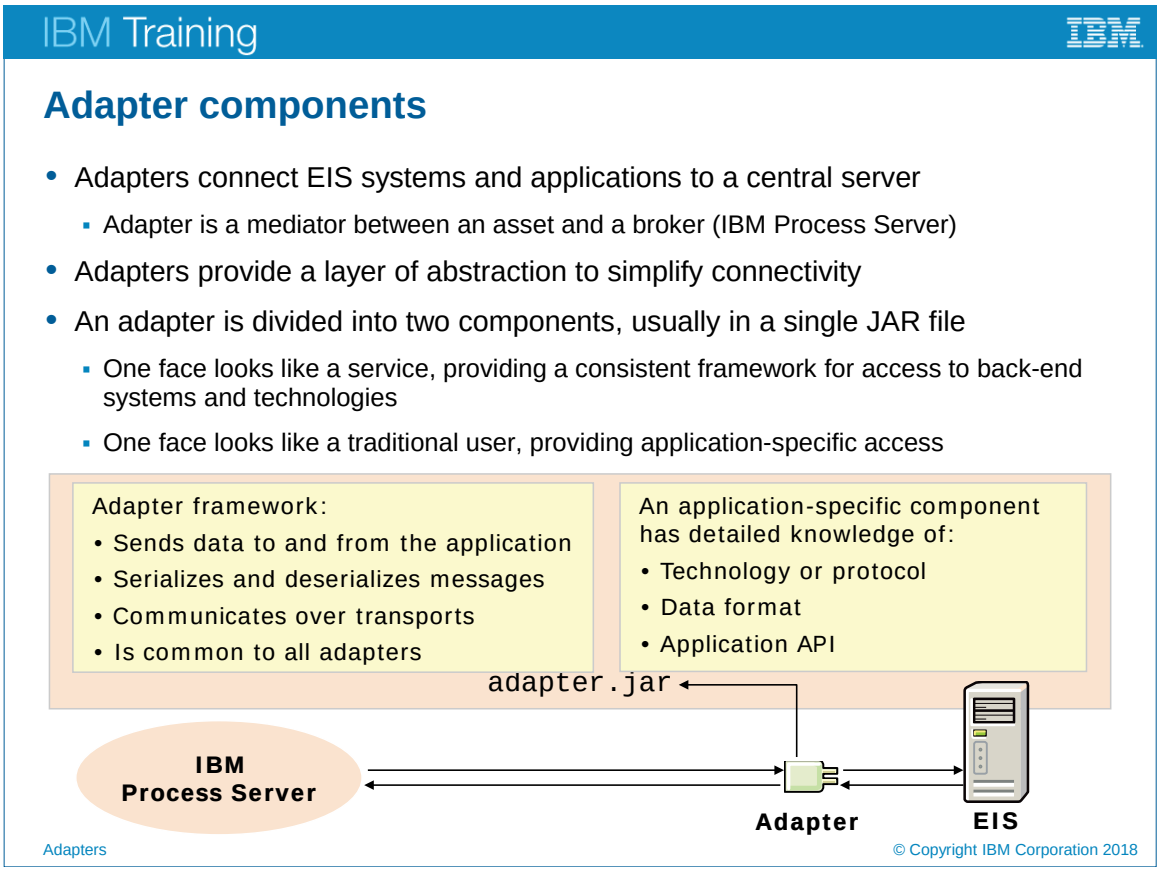
Adapters

Introduction to adapters

Adapters also provide other services, such as CEI event monitoring, transaction support, asynchronous communication, connection pooling, security (beyond web service), and a discovery utility for generating interfaces.

As you design and assemble your SOA solutions, on many occasions you must integrate SOA applications with an existing IT asset. Ideally, you are able to access that asset as a service just as you access other components. If the asset does not have a service interface, examine the environment to see whether the service interface can be added easily. On some occasions, the environment cannot directly support a service interface. You can use adapters to access and include such assets in your SOA solutions.

Organizations large and small invested in enterprise information systems over the past several decades, and as such are critically dependent upon their continued execution. Invariably any organization has a large proportion of its data that one, or more typically, many EIS systems manage. These systems fall into categories, such as ERP systems like SAP, and database systems like Oracle or DB2. Access to the data that these applications manage is provided through a set of exposed services, but adapters can also provide access to these services.



Adapter components

An adapter acts as a mediator between your SOA world and the asset. Access to the asset is not achieved directly. A component that is called an adapter presents a service interface to the SOA components, providing the service that the asset must expose. This interface is one face of the adapter. The other face of the adapter communicates directly with the asset by using whatever technology is appropriate for that asset. This communication can be achieved through proprietary APIs or various other techniques.

Adapters are specific to the version of the application for which they are written. If the application changes its API, a new adapter is developed. However, if the application data schema changes, the adapter code is not required to be changed. An advantage of the adapter framework is that when coding an adapter, it is not necessary for the developer to know the means of connecting with the message queues or configuration files. The framework automatically handles that connection. Instead, the developer writes the interaction only between the application programming interface and the adapter API methods to process business objects.

The adapter is written in the language that the API is written in, for example, SAPAdapter.jar written in Java or ClarifyAdapter.dll written in C/C++.

A toolkit is available for the creation of custom WebSphere Business Integration and JCA adapters. The framework for WebSphere Business Integration Adapters is proprietary but common among all WebSphere Business Integration Adapters.

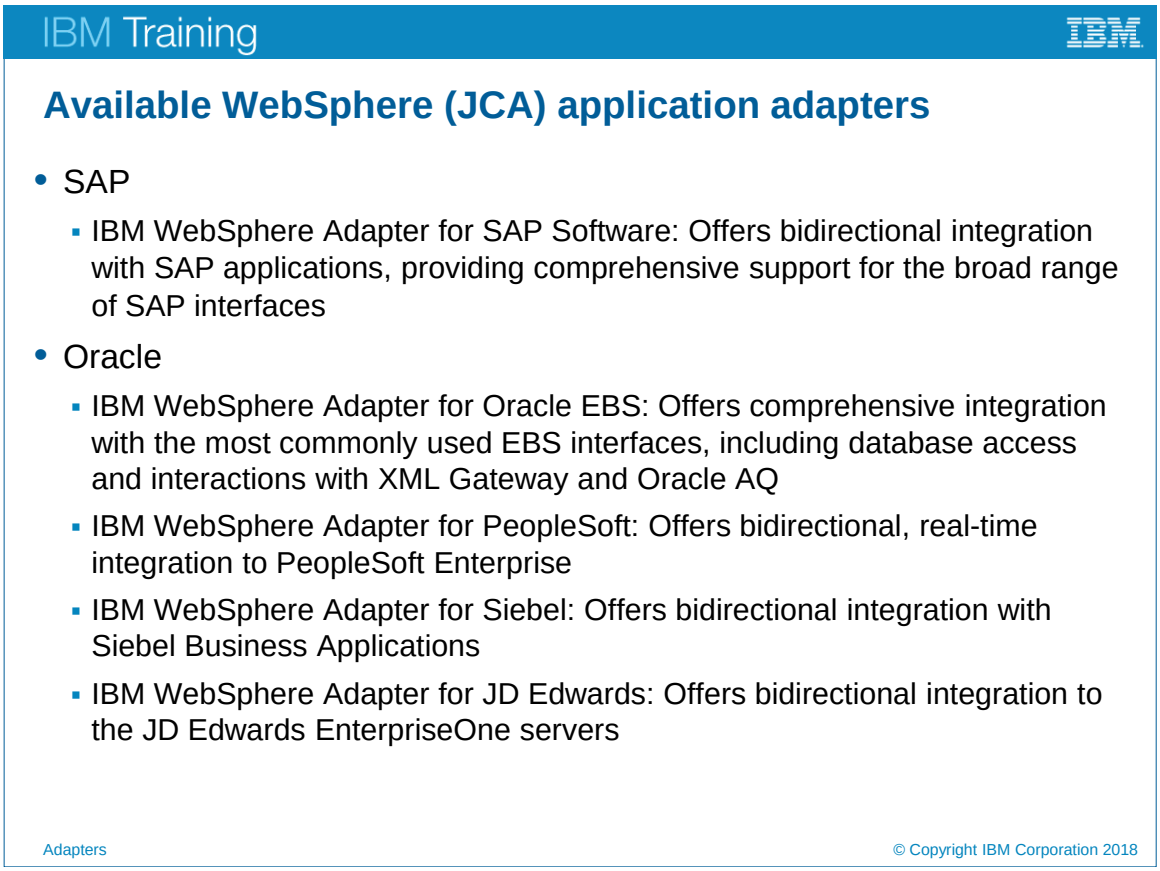
Adapters are lightweight, distributed, metadata-driven components that are ready to run. They require no code modification, only configuration and transaction metadata in the form of business object definitions. One adapter can handle an object with multiple operations and multiple objects. Furthermore, business rules, transformation, and routing are delegated to the integration server: they are outside the scope of the adapter, which provides a common means for an integration server to interact with the application. The primary benefits are rapid deployment, ease of maintenance, and flexible distributed deployment.

IBM Training

WebSphere (JCA) Adapters

- WebSphere Adapters accelerate integration projects with a set of Java EE Connector Architecture (JCA) capabilities that service-enable your assets
  - Help in minimizing the need for integration coding and create standard interface points
- Two types of adapters: application and technology
- Application adapters connect to existing packaged applications so you can use data and services specific to the applications
  - All JCA application adapters are included in IBM Integration Designer for development use in the unit test environment (UTE)
- Technology adapters provide connectivity to data through technologies and protocols
  - All WebSphere Technology Adapters (JCA only) included in IBM Integration Designer are for development and production use with licensed copies of IBM Process Server

WebSphere (JCA) Adapters



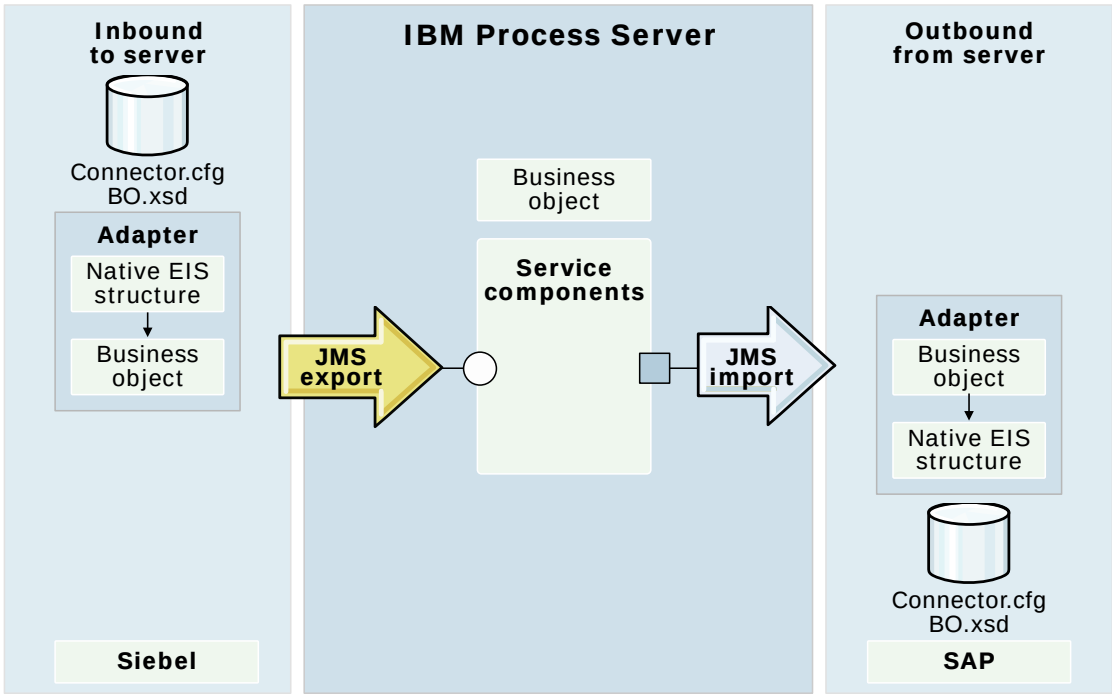
Available WebSphere (JCA) application adapters

IBM WebSphere Adapters accelerate integration projects with a set of Java Connector Architecture (JCA) capabilities that service-enable your assets. They help you minimize the need for integration coding and create standard interface points. WebSphere Adapters help extend service-oriented architecture (SOA) applications beyond organizational walls to customers, partners, and suppliers. They can reduce maintenance and development costs while optimizing and renewing the value of your enterprise assets.

For more information on application adapters, go to:
http://www.ibm.com/software/products/en/adapters

IBM Training

Architecture



Adapters

Architecture

IBM Training

Available WebSphere (JCA) technology adapters

- IBM WebSphere Adapter for Flat Files
  - Bidirectional integration through the exchange of local flat files
- IBM WebSphere Adapter for JDBC
  - Bidirectional JDBC data source integration of tables, views, and stored procedures
- IBM WebSphere Adapter for FTP
  - Bidirectional exchange of remote flat files with support for FTP over SSL
- IBM WebSphere Adapter for Email
  - Bidirectional integration through exchange of email by using mail systems
  - Inbound support for IMAP and POP3, and outbound support for SMTP
  - Support for transformation of content or attachments, and the local mail archive
- IBM WebSphere Adapter for IBM System i
  - Provides bidirectional integration to native i5/O applications, including RPG programs on the IBM i platform
- IBM WebSphere Adapter for Lotus Domino
  - Bidirectional exchange of business data (calendar entry, task list item, or note) between SCA applications and Domino servers)

Adapters

Available WebSphere (JCA) technology adapters

IBM WebSphere Adapter for Technology Standard delivers connectivity by using technology standards, protocols, and database. It uses the Java Connector Architecture (JCA) and enterprise metadata discovery specifications to provide integration with graphical discovery tools without writing code.

For more information on technology adapters, go to: http://www.ibm.com/software/products/en/websphere-adapter-for-technology

IBM Training

Advantages of Java EE Connector Architecture (1 of 2)

- JCA provides a Java open standard for EIS connectivity
- JCA provides added qualities of service
- Connection management
  - Application server JCA container maintains a reusable connection pool
  - Provides a scalable application environment that can support many EIS clients
- Transaction management for outbound transactions
  - Distributed transactions (through XA)
  - Local transactions
  - Support is dependent upon EIS
- Inbound transactions: Assured event delivery
  - Ensures “once and only once” delivery of events (requires acknowledgment)
  - Works even if EIS does not support transactions (through event staging)
- Security management

Advantages of Java EE Connector Architecture

The JCA V1.5 specification has several advantages. First, it is an industry standard (which different vendors widely accept). Second, by moving into the Java EE environment, more quality of service qualifiers are provided through the WebSphere Resource Adapter container.

JCA provides a non-proprietary framework for creating a Java EE adapter component. JCA is a Java EE standard that was created to normalize the way Java applications access enterprise applications. Customers can use the JCA resource adapter to integrate business functions and to provide a centralized source for the application resource adapter development. A resource adapter is the component of the JCA that provides EIS connectivity.

The following list contains more characteristics of the JCA specification:

- **Connection pooling**: JCA container maintains a pool of connections
- **Security**: End-to-end Java EE security model
- **Transaction support** (the following are supported):
  - Outbound:
    - XA (distributed transactions)
    - Local transactions
  - Inbound (assured event delivery):
    - Ensures “once and only once” delivery of events
    - Works even if EIS does not support transactions (through an optional event staging database)

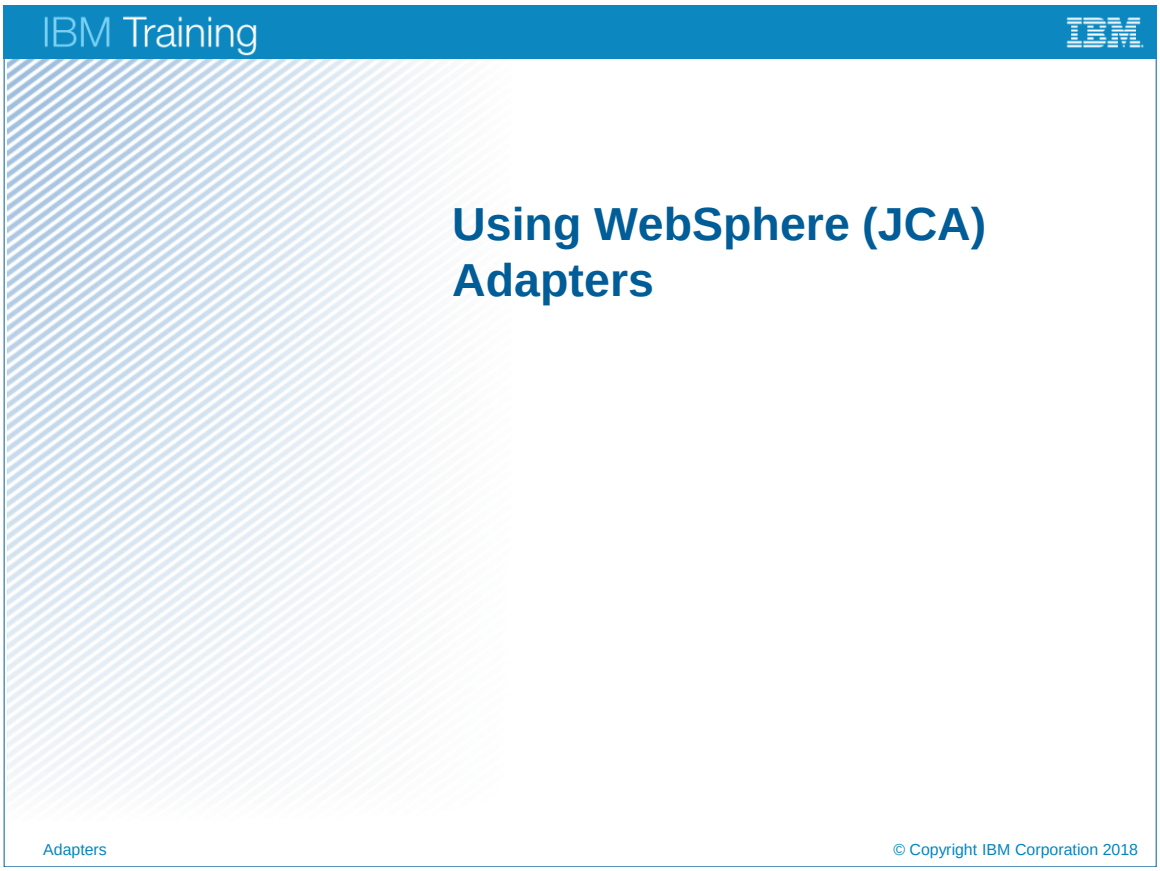
IBM Training

Advantages of Java EE Connector Architecture (2 of 2)

- Open standard provides faster time-to-market and lower maintenance
  - JCA designed for adoption by independent software vendors
- JCA adapters are multithreaded and support two kinds of event delivery
- Ordered delivery:
  - Single thread delivers one event at a time
  - Events arrive in the order they were created
  - Event ordering takes precedence over throughput
- Unordered delivery:
  - Multiple threads deliver events on several channels simultaneously
  - Event order is not kept
  - Higher throughput takes precedence over event order

Adapters

If the delivery type is set to “ORDERED,” an instance of the adapter runs in a single thread that delivers the events in the order that they were created. If the delivery type is set to “UNORDERED,” events get delivered simultaneously and the delivery order is not guaranteed. The benefit of “UNORDERED” is higher throughput. The default is “ORDERED.”



Using WebSphere (JCA) Adapters

IBM Training

Using WebSphere Adapters

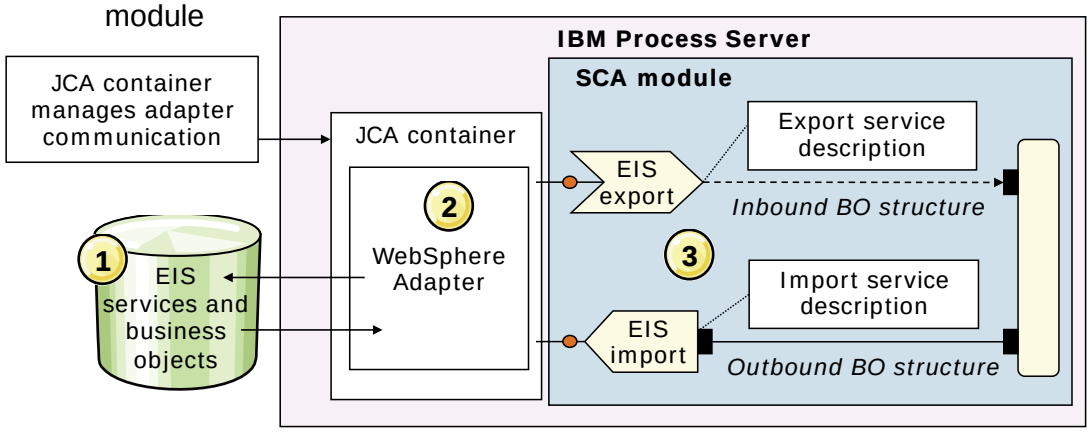
- The External Service wizard is used to discover connectivity information about an application or technology
- WebSphere Adapters use Enterprise Metadata Discovery (EMD) inside the External Service wizard to introspect an EIS
  - EMD defines common API adapter uses to expose EIS services and business objects to application development tools
  - Single tool supports multiple EIS systems and multiple adapters
  - EMD is provided as an IBM Integration Designer plug-in
- WebSphere Adapters use the Data Exchange Service Provider Interface (DESPI) to transfer data between adapter components and the IBM Process Server runtime
  - DESPI is an IBM extension to JCA
  - It is used to transfer data between adapters and various runtimes
  - Provides runtime data format independence

Using WebSphere Adapters

IBM Training

Enterprise Metadata Discovery

- When the External Service wizard is run:
  1. EMD introspects the EIS
  2. User selects entities of interest (interfaces or business objects) for use with the adapter project
  3. EIS bound imports and exports, WSDL interfaces, and business object definitions are generated and packaged for deployment in the SCA module



Adapters

Enterprise Metadata Discovery

Enterprise Metadata Discovery provides a rich environment for building and managing enterprise class applications and solutions. With the rise of SOA, more enterprises are deploying SOA solutions in a Java EE environment, but find it necessary to service-enable their existing IT assets to realize the full benefits of business automation. IBM and BEA collaborated on specifications that provide more productive development and deployment of this “last stage” of integration. The “Enterprise Metadata Discovery Specification for J2CA” is a cross-industry initiative started by IBM and BEA, and supported by adapter vendors and application vendor partners (it is not part of the original JCA specification). This new specification provides a standard way of managing this “last mile” with a seamless design-time experience (by using a top-down approach), thus allowing customers significant productivity improvements in service-enabling their existing IT assets.

The Enterprise Metadata Discovery specification intends to solve the problem of standard rich tool interfaces for adapters. Thus it unlocks the potential of the integration industry to deliver higher-quality adapters to more end systems at lower costs. Adapter vendors can focus on building more and better adapters, without worrying about interoperability of proprietary extensions. Infrastructure vendors can focus on building better platforms and better development experiences. Application vendors can build their own “last mile” components, confident that a fully standard implementation addresses both runtime and design-time interoperability requirements. As a result, any adapter product can plug seamlessly into any Java EE SOA implementation, delivering both superior end-to-end runtime behavior and developer productivity.

The result to customers is greater adapter availability and quality at a lower cost. Existing IT assets can be a service that is enabled more readily, thus reducing the total cost of ownership and improving time-to-value of an SOA implementation. Moreover, each component in the end-to-end solution is built to agreed-upon specifications as opposed to proprietary extensions, thus reducing vendor lock-in.

For more information about the EMD specification, see:
http://www.ibm.com/developerworks/webservices/library/ws-soa-eisjca/#EMD

The EMD tool (through the external service wizard in IBM Integration Designer) does the following tasks:

1. The adapter metadata discovery service introspects the EIS and examines its data structure.
2. Based on what is discovered, the EMD tool generates definitions for business objects, imports and exports, and WSDL interfaces.

When an adapter is used, the module that contains the adapter has an export or import component that is bound to the adapter. The WebSphere Application Server JCA container manages the communication between the adapter and the enterprise application or technology.

Binding is done at three levels:

- **Interface binding**: Describes the mapping between the EIS service connection information and the service interface
- **Method binding**: Expresses the relationship between interface operations and service interaction or event information
- **Data binding**: Defines how the service input or output data is mapped to the native data as understood by the EIS service

For SCA clients, the function of the adapter is exposed through a WSDL interface. An SCA client invokes the adapter service through the `EISImportBinding`, which EMD produces. The inbound service description is an `EISExportBinding`, which EMD also produces. EMD generates business objects, which both SCA components and the adapter use. Using SCA components and business objects to interact with adapters fits the goals and the vision of SOA solutions.

IBM Training

Discovered interactions

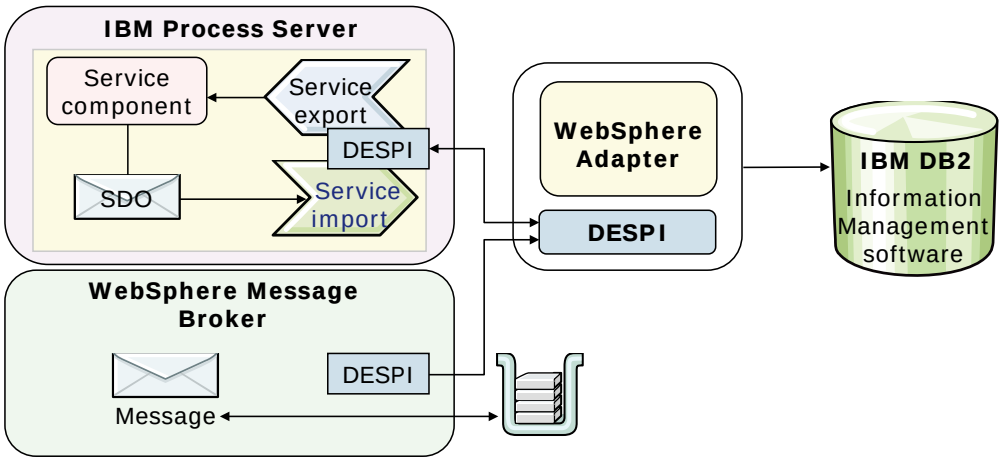
- The EMD tool discovers the interaction styles of the EIS based on your input in the External Service wizard
- Two types of interaction styles:
  - Outbound: Client initiated
  - Inbound: EIS initiated
- Two modes of interaction for each style: Request/response or one-way
- Request/response interaction
  - Request/response interaction takes a request and returns a response
  - Generated method includes both an input and an output
- One-way interaction
  - Takes a request, but does not return a response
  - Generated method includes only an input

Discovered interactions

IBM Training

Data Exchange Service Provider Interface

- Adapters handle data internally, in a format-independent manner, by using DESPI
  - An export or import data binding produces and consumes business objects and communicates with the adapter by using the Data Exchange SPI
  - The DESPI interface decouples the adapter from data representations that the EIS and the business object data model understand



Adapters

Data Exchange Service Provider Interface

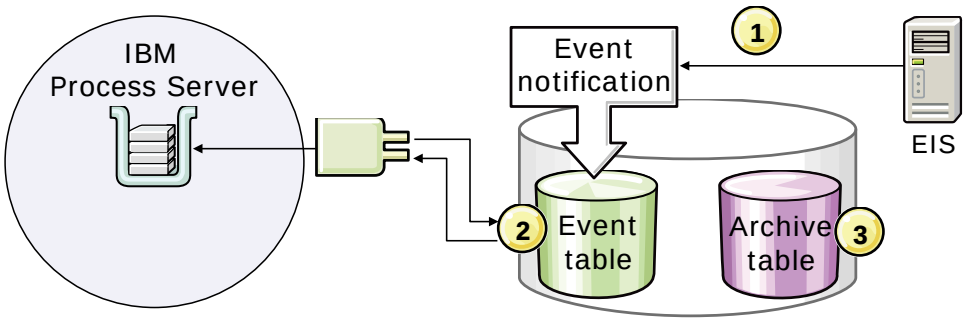
DESPI is the interface that resource adapters and runtime components use to exchange business object data. The Data Exchange SPI architecture, which is based on the concept of cursors and accessors, abstracts the data type so that an adapter can be written only once. It can then work on runtime environments that support different data types, such as data objects and JavaBeans. A DESPI architectural diagram is included in the appendixes.

IBM Training

Application adapters and event polling

Most application adapters require event polling in the EIS

1. A database trigger or script creates an event in the event table
2. The adapter polls the event table for new items
   - A business object that is the subject of the event is retrieved
   - The business object is placed in a queue for processing
3. When processed, the adapter moves the event from the event table to the archive table (for auditing)
   - Archive table cleanup is done manually by using provided scripts



Adapters

Application adapters and event polling

JCA application adapters often require modification of the EIS so that the application generates an “adapter-identifiable” event. Modification includes:

- Installation of scripts
- Creation of database triggers and tables
- Addition of objects and tools

IBM adapters have components and scripts to build the schema and tables that the adapter needs for event notification on a particular application. If an adapter is publishing events, the corresponding application needs the following items:

- A storage location (an event table) in which to cache key event data (a database table or directory); event table polling speed is configurable
- A mechanism to capture changes in the application and create an event (triggering mechanism: database trigger, script, or workflow)
- An archive location to note events that are successfully sent to the integration tool (database or directory)

IBM Training

Unit summary

- Describe the purpose and business value of using adapters in applications
- Describe the capabilities of WebSphere (JCA) adapters
- List the advantages of using the JCA architecture for WebSphere Adapters
- Describe how to use the External Service wizard for WebSphere Adapters

Unit summary

IBM Training

Checkpoint questions

1. True or False: Adapters provide integration with enterprise information systems (EIS) that have clearly defined service interfaces.
2. True or False: The External Service wizard discovers and generates the necessary SCA artifacts, including business objects, interfaces, and import and export components.
3. True or False: Technology adapters connect to existing packaged applications so you can use data and services specific to the applications.

Checkpoint questions

IBM Training

Checkpoint answers

1. False. One of the key advantages of adapters is that they provide integration with enterprise information systems (EIS) that do not provide service interfaces.
2. True.
3. False. Application adapters connect to existing packaged applications so you can use data and services specific to the applications.

Checkpoint answers

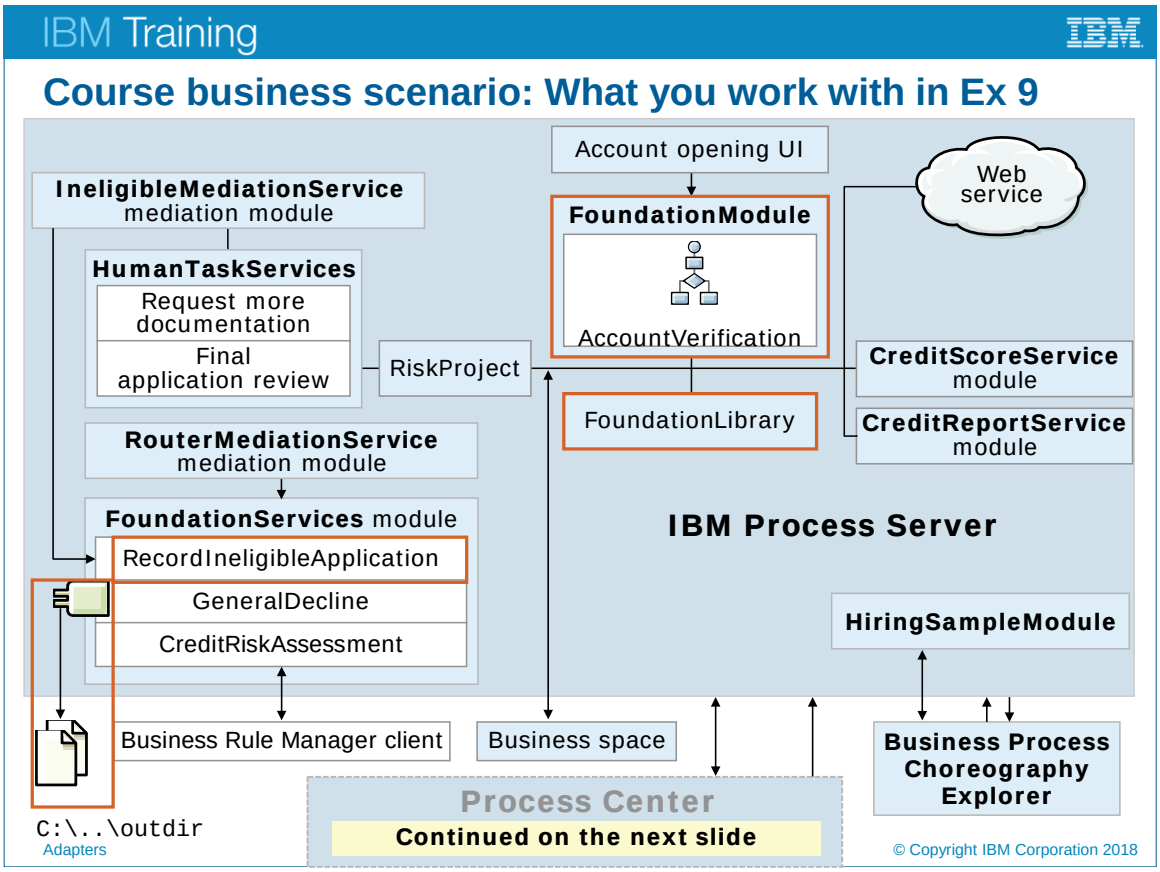
IBM Training

Exercise 9: Implementing WebSphere (JCA) adapters

After completing this exercise, you should be able to:

- Configure the WebSphere Adapter for Flat Files
- Use the external service tool to generate artifacts that are used in an application
- Incorporate adapter-related SCA artifacts in an assembly diagram
- Test an adapter in the IBM Integration Designer test environment

Exercise 9: Implementing WebSphere (JCA) adapters



Course business scenario: What you work with in Ex 9

IBM Training

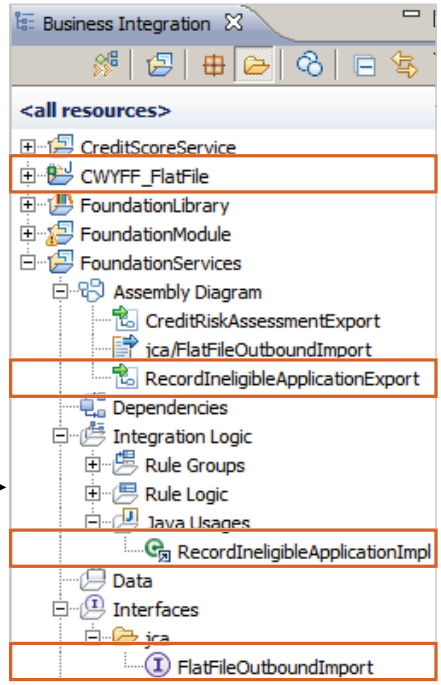
Components that are required for Exercise 9 (1 of 2)

Prebuilt components that are imported in the lab:

1. **FoundationModule**
2. **CreditScoreService**
3. **FoundationLibrary**
4. **FoundationServices** that you built in Unit 10

New components that you create in this lab:

1. **CWYFF_FlatFile** connector project
2. **FlatFileOutboundImport** import component
3. **RecordIneligibleApplicationExport** export component
4. **RecordIneligibleApplicationImpl.java** component



Adapters

Components that are required for Exercise 9

IBM WebSphere Adapters make it possible for Java Platform, Enterprise Edition (Java EE) components, such as applications, to communicate with enterprise information system (EIS) resources. An EIS is the information infrastructure for an enterprise (for example, an enterprise resource planning system). A WebSphere adapter acts as an intermediary between the Java EE component and the EIS. This way, it is not necessary for the Java EE component to understand the low-level API or data structures of the EIS.

The IBM WebSphere Adapter for Flat Files facilitates the exchange of business data in the form of delimited records between file systems and Java EE applications. The adapter supports inbound and outbound operations and the use of business objects, business components, and business services.

In this exercise, you implement a service to record ineligible applications as XML files on the file system. This service uses the WebSphere Adapter for Flat Files. The AccountVerification process invokes the service when the eligibleApplication attribute is set to false.

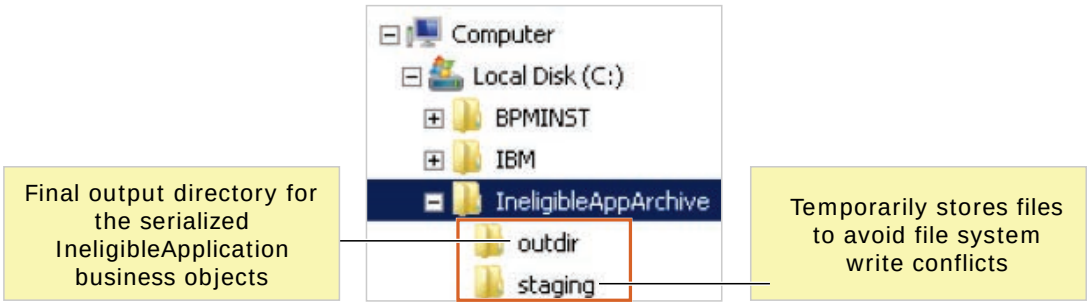
See the process application model that was created previously. In this exercise, you implement the Record Ineligible Application activity. The purpose of this activity is to record ineligible applications as XML files on the file system. This service uses the WebSphere Adapter for Flat Files.

IBM Training

Components that are required for Exercise 9 (2 of 2)

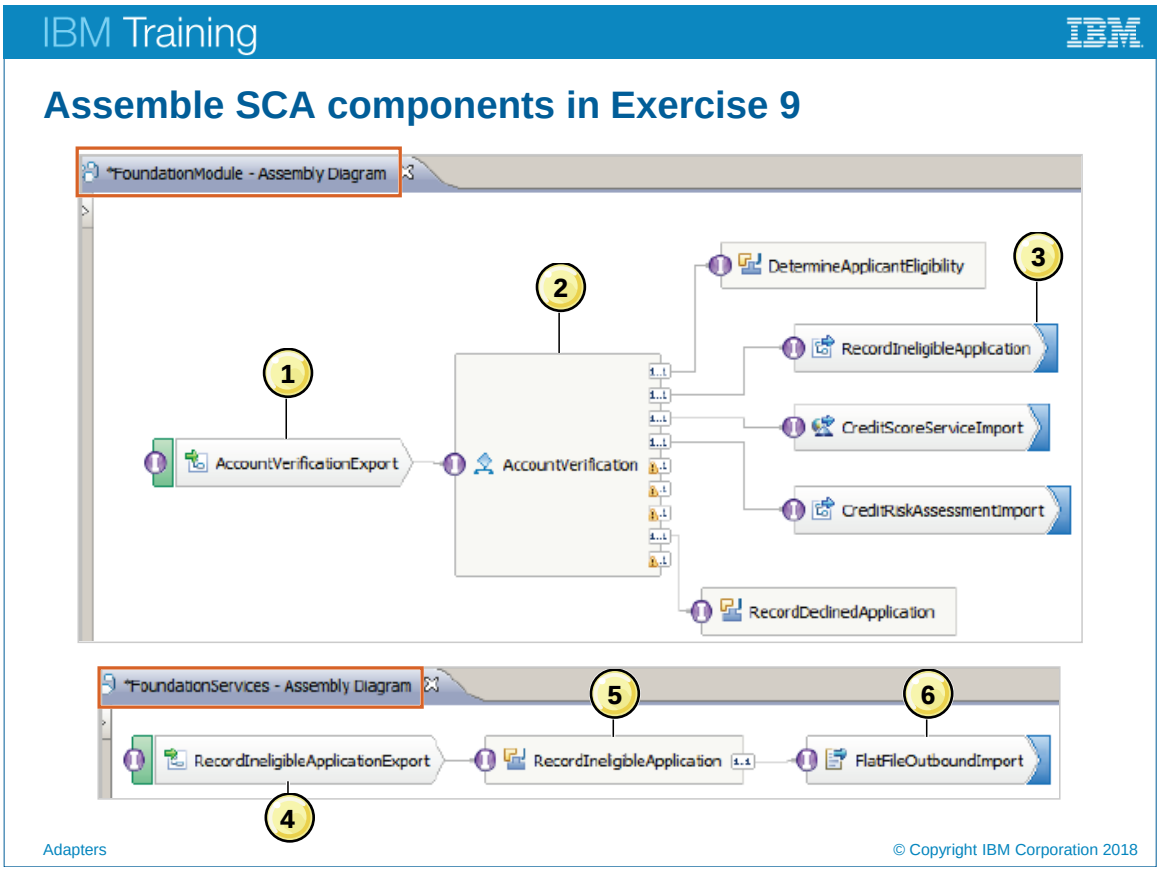
Two new directories that you create in this lab:

1. The **`C:\IneligibleAppArchive\staging`** directory
2. The **`C:\IneligibleAppArchive\outdir`** output directory
   - The IneligibleApplication business object is written to this output directory



Adapters

You run the External Service wizard to configure the WebSphere Adapter for Flat Files. The adapter uses the directories that you examined previously. The adapter code is deployed inside the FoundationServices module, and a FlatFileOutboundImport component is created on the FoundationServices assembly diagram. The FlatFileOutboundImport component invokes the recordIneligibleApplication operation to write an `IneligibleApplication.txt` file to `C:\IneligibleAppArchive\outdir`.



Assemble SCA components in Exercise 9

Finally, after creating the flat file adapter component, you assemble the SCA components in IBM Integration Designer and then test the application.

The following steps are illustrated in the diagram:

1. The AccountVerificationExport component exposes the AccountVerification business process.
2. When the application is ineligible, the AccountVerification process needs to record the ineligible application in the database and terminate the process. It calls the RecordIneligibleApplication import component.
3. The RecordIneligibleApplication import component is used to call the application or service outside FoundationModule. In this scenario, it calls the RecordIneligibleExport component in the FoundationServices module.
4. The RecordIneligibleExport component calls the RecordIneligibleApplication Java component.

5. The RecordIneligibleApplication component is a Java component that creates the output message.

   “`Account verification recorded this application as ineligible for the customer <company name>`” is going to be recorded to the system. In this scenario, it calls the FlatFileOutboundImport component.

6. The FlatFileOutboundImport component writes the output message to the file system. In this scenario, it writes to a text file in the `C:\IneligibleAppArchive\outdir` output directory.

Exercise 9: Implementing WebSphere (JCA) adapters

Purpose:
In this exercise, you configure a WebSphere adapter as part of a business process solution.

IBM WebSphere Adapters make it possible for Java Platform, Enterprise Edition (Java EE) components, such as applications, to communicate with enterprise information system (EIS) resources. An EIS is the information infrastructure for an enterprise (for example, an enterprise resource planning system). A WebSphere adapter acts as an intermediary between the Java EE component and the EIS. This way, it is not necessary for the Java EE component to understand the low-level API or data structures of the EIS.

Requirements

Completing the exercises for this course requires a lab environment. This environment includes the exercise support files, IBM Process Designer, IBM Process Portal, IBM Process Center, and IBM Integration Designer test environment.

Exercise instructions

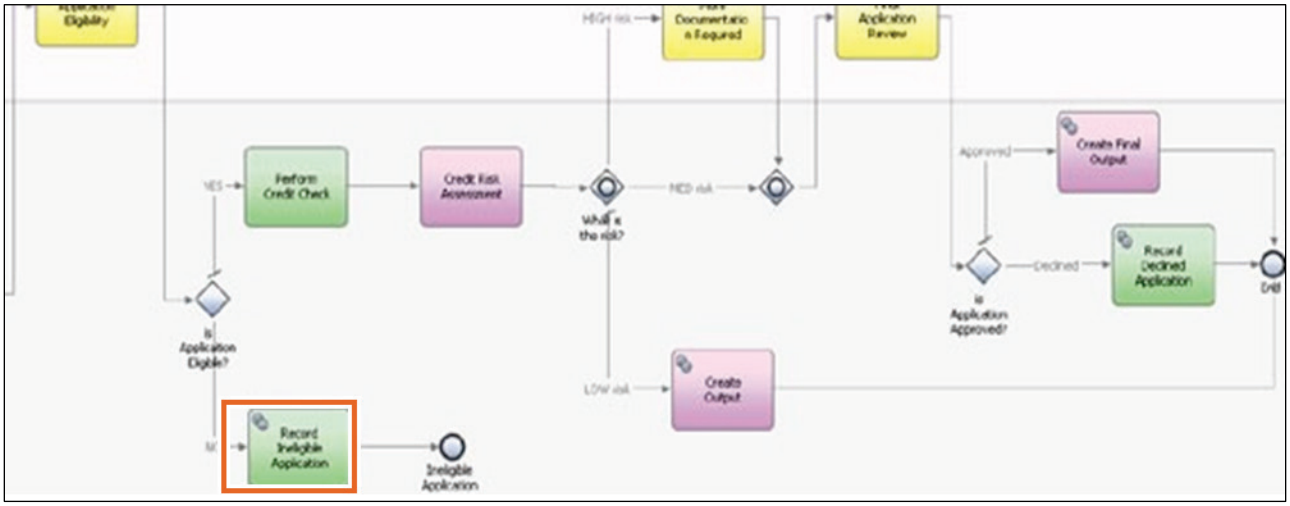
The IBM WebSphere Adapter for Flat Files connects Java Platform, Enterprise Edition components that are running on WebSphere products (such as IBM Process Server or WebSphere Enterprise Service Bus) with file systems. For example, the Java EE component, when configured to work with the adapter, can create a file with specified contents on the file system. The file can then be accessed from the file system by another application.

The IBM WebSphere Adapter for Flat Files facilitates the exchange of business data in the form of delimited records between file systems and Java EE applications. The adapter supports inbound and outbound operations and the use of business objects, business components, and business services.

In this exercise, you implement a service to record ineligible applications as XML files on the file system. This service uses the WebSphere Adapter for Flat Files. The AccountVerification process invokes the service when the eligibleApplication attribute is set to `false`.

See the process application model that was created previously. In this exercise, you implement the Record Ineligible Application activity. The purpose of this activity is to record ineligible applications as XML files on the file system. This service uses the WebSphere Adapter for Flat Files.

Do not be concerned about reading the small text in this diagram. The purpose of the solution diagram is to view the connection wiring and the flow.



Part 1. Configure the WebSphere Adapter for Flat Files.

In this portion of the exercise, you use the external service wizard to configure the flat file adapter for outbound processing. Though your application needs only outbound, the adapter supports both inbound and outbound communication with the file system.

1. Open the Exercise 9 workspace.

1. On your desktop, open the **Exercise Shortcuts** folder.
2. Double-click the **Exercise 9** shortcut.

   Allow Integration Designer a few moments to build the workspace. You can view the workspace build status at the lower-right corner of Integration Designer. Wait until the status reaches 100%, at which point the workspace is built, and the status progress bar disappears.
3. If you get a message that the server is already set to publish, then click **OK**.

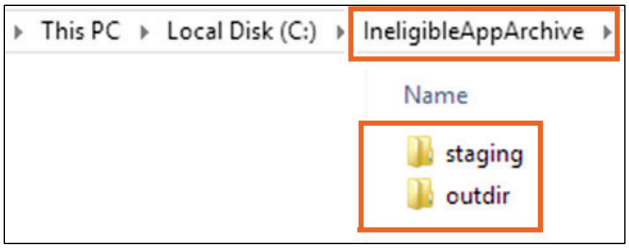
   If the server is already running from the previous exercise, you get this message.
4. Close the **Getting Started** tab.

2. Create two file system directories for the flat file adapter.

The adapter requires a staging directory `C:\IneligibleAppArchive\staging`, and an output directory, `C:\IneligibleAppArchive\outdir`. When configured, the adapter writes the contents of the IneligibleApplication business object into the `C:\IneligibleAppArchive\outdir` directory.

1. Open **Windows Explorer** and browse to the `C:\` directory.

2. Create a directory that is named: `IneligibleAppArchive`
3. Create two subdirectories, named: `outdir` and `staging`



The `\staging` directory is used to temporarily store files to avoid file system write conflicts. The `\outdir` directory is the final output directory for the serialized IneligibleApplication business objects.

4. Close Windows Explorer.

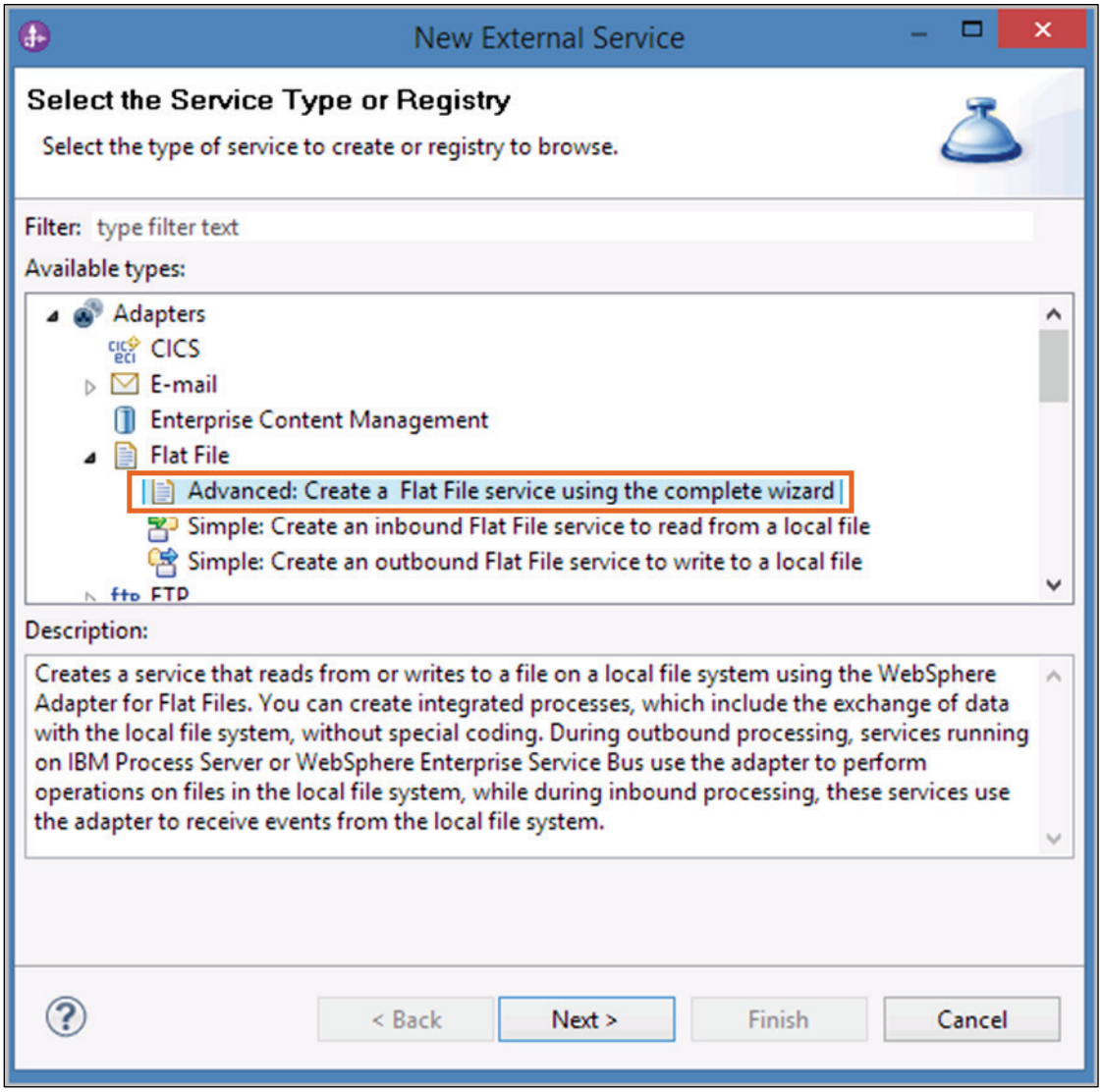
Part 2. Use the external service tool to generate artifacts that are used in an application.

1. Run the External Service wizard to configure the WebSphere Adapter for Flat Files.

The adapter uses the directories that you examined previously. The adapter code is deployed inside the FoundationServices module, and a FlatFileOutboundImport component is created on the FoundationServices assembly diagram. The FlatFileOutboundImport component invokes the recordIneligibleApplication operation to write an `IneligibleApplication.txt` file to `C:\IneligibleAppArchive\outdir`.

1. In the Business Integration view, right-click the **FoundationServices** module, and click **New > External Service** from the menu.

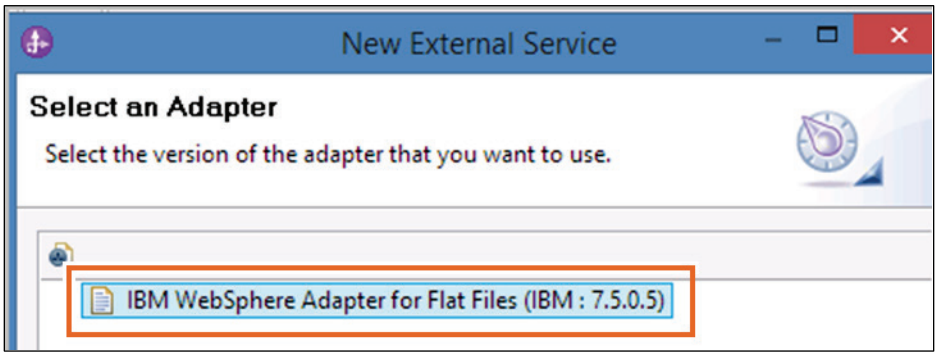
2. In the “Select the Service Type or Registry” window, in the Available types section, expand **Adapters > Flat File** and select **Advanced: Create a Flat File service using the complete wizard**.



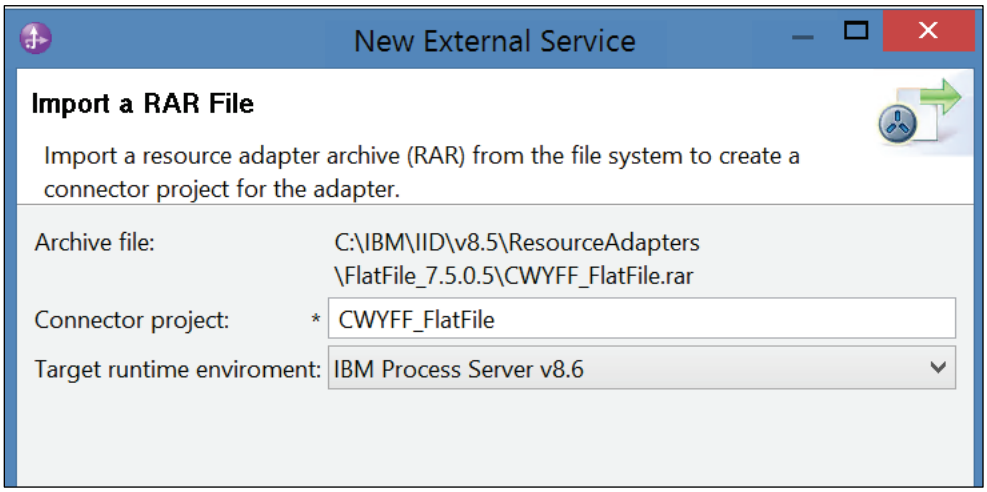
The “Simple” options use sample patterns to generate adapter services. For more information about patterns in IBM Integration Designer, see the product documentation.

3. Click **Next**.

4. In the “Select an Adapter” window, select **IBM WebSphere Adapter for Flat Files (IBM: 7.5.0.5)**.



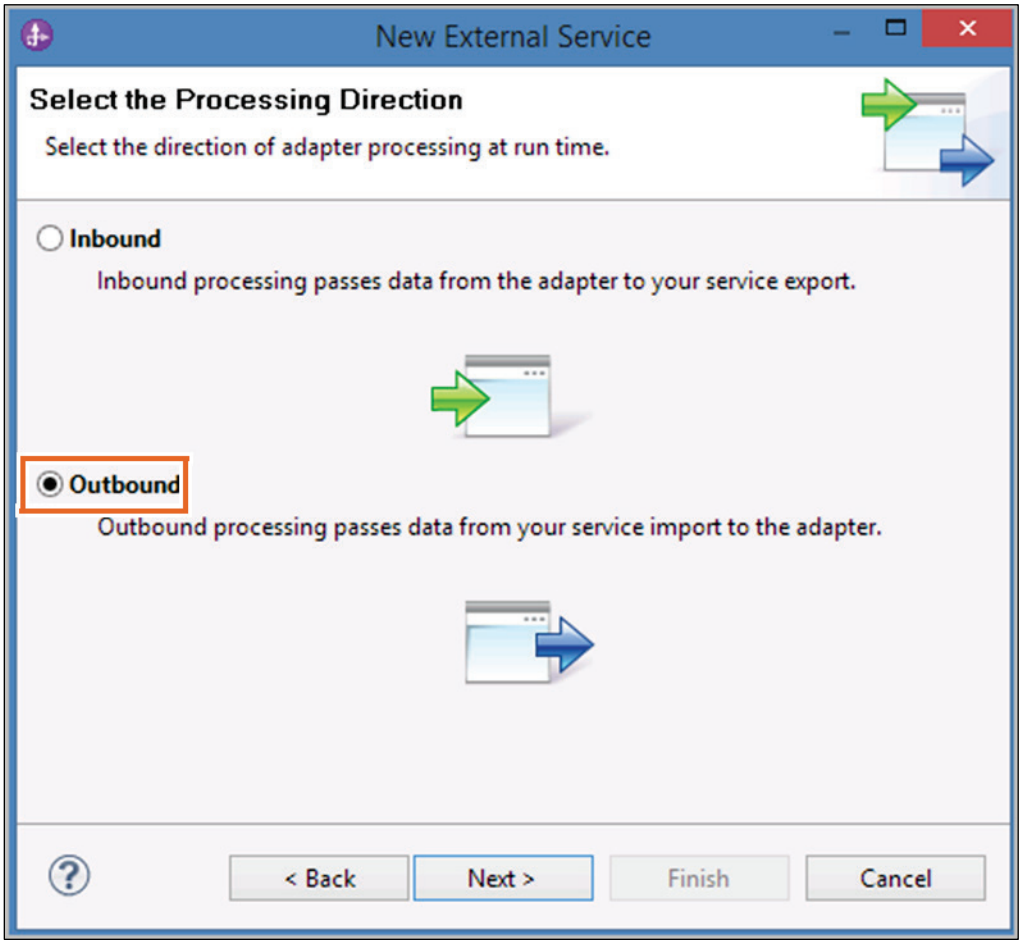
5. Click **Next**.
6. In the “Import a RAR file” window, accept the default options. The **Connector project** field is set to `CWYFF_FlatFile`, and the **Target runtime environment** field is set to **IBM Process Server v8.6**.



Note: A RAR file is a Java resource adapter archive file that is used to package a resource adapter for the Java 2 Connector (J2C) architecture. The RAR file for an adapter can be deployed inside an application or at the server level for use by multiple applications. For information about deploying adapters independently, see the product documentation.

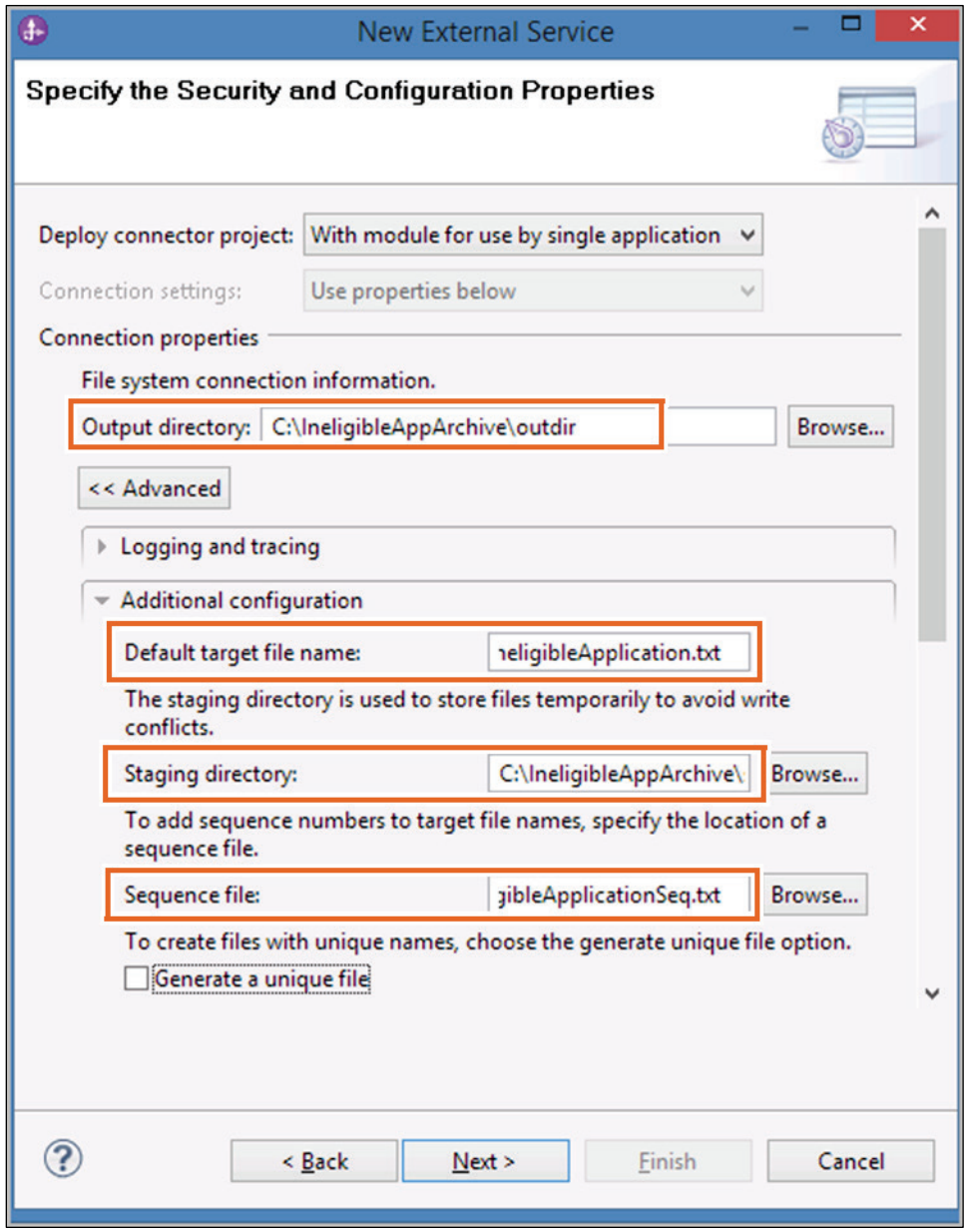
7. Click **Next**.

8. In the “Select the Processing Direction” window, select **Outbound**.



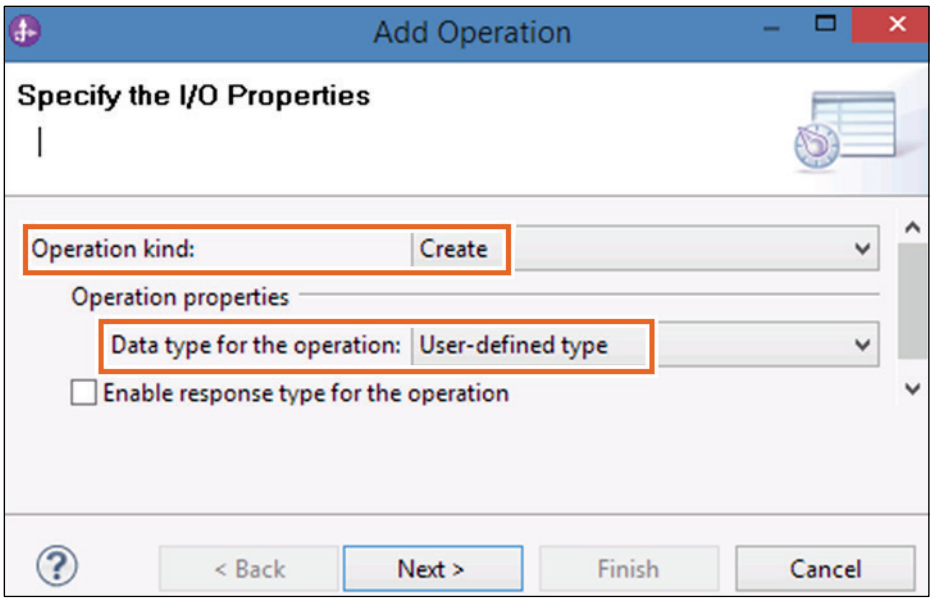
9. Click **Next**.
10. In the “Specify the Security and Configuration Properties” window, enter the following information.
    - Verify that **Deploy connector project** is set to: `With module for use by single application`
    - For **Output directory**, click **Browse**, go to `C:\IneligibleAppArchive\outdir`, and click **OK**.
    - Click **Advanced** to open the additional connection properties, expand **Additional Configuration**, and enter the following information:
        - In the **Default target file name** field, type: `IneligibleApplication.txt`
        - Click **Browse** for **Staging directory**, go to `C:\IneligibleAppArchive\staging`, and click **OK**.

- In the **Sequence file** field, type the following path and file name:
  `C:\IneligibleAppArchive\outdir\IneligibleApplicationSeq.txt`

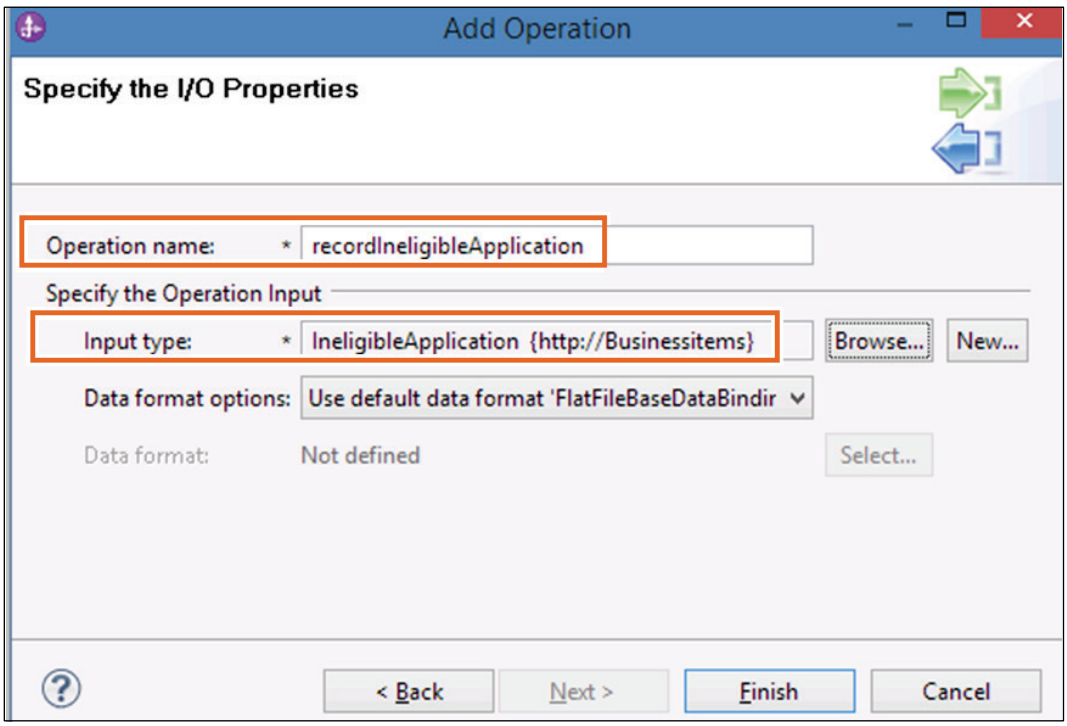


11. Accept the remaining default options and click **Next**.
12. In the “Add, Edit or Remove Operations” window, click **Add** to add an operation.

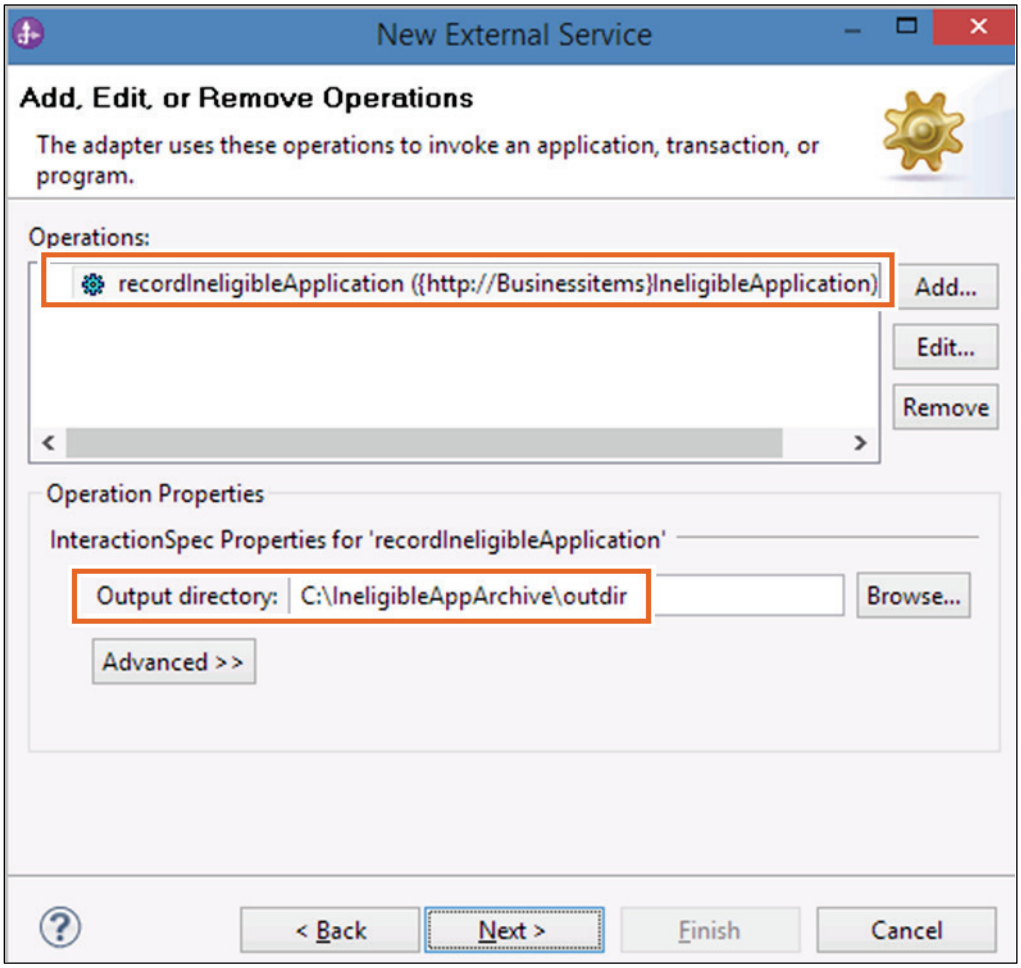
13. In the “Specify the I/O Properties” window, verify that **Operation kind** is set to `Create` and verify that **Data type for the operation** is set to: `User-defined type`



14. Accept the remaining default options and click **Next**.
15. In the second “Specify the I/O Properties” window, enter the following information.
    - In the **Operation name** field, type: `recordIneligibleApplication`
    - Click **Browse** beside the **Input type** field, select the **IneligibleApplication** business object, and click **OK**.

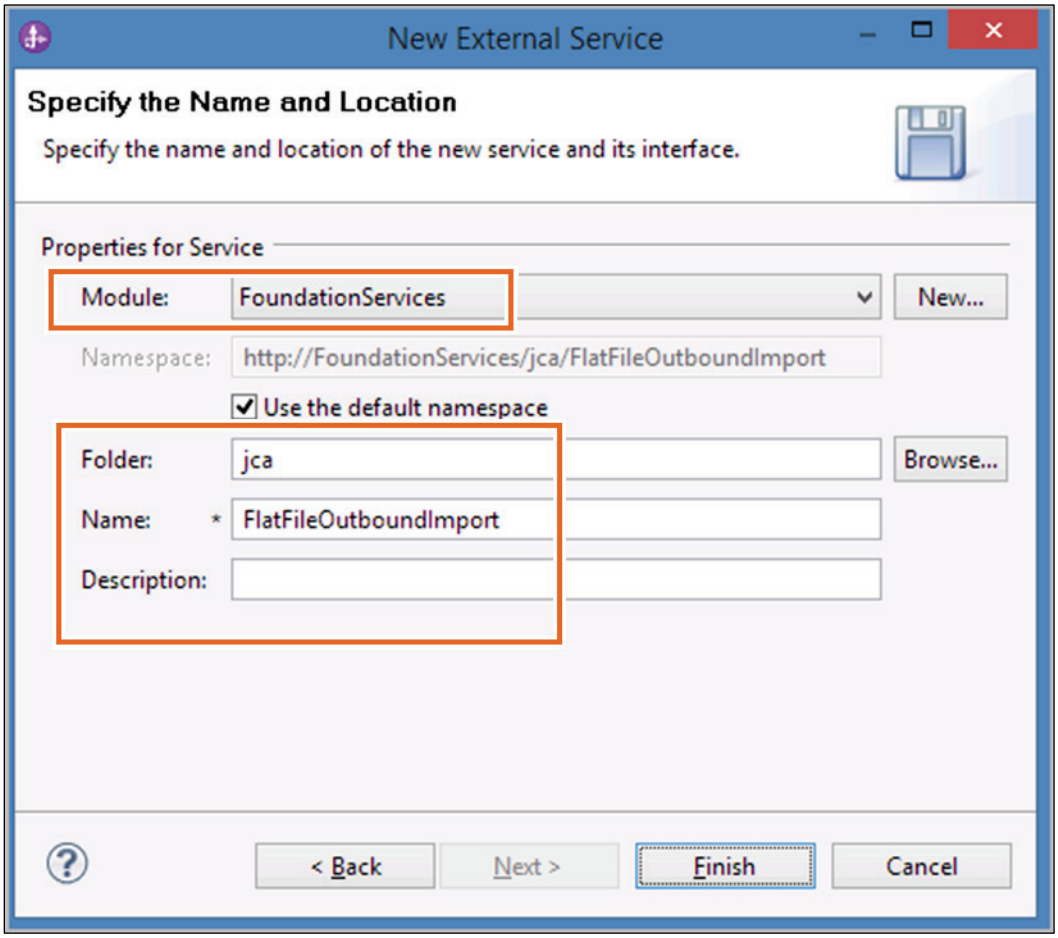


16. Accept the remaining default options and click **Finish**.
17. When you are returned to the “Add, Edit or Remove Operations” window, provide the following information:
    - Verify that `recordIneligibleApplication` is added to the Operations window.
    - In the **Output directory** field, type `C:\IneligibleAppArchive\outdir` or use **Browse** to locate the directory.



18. Accept the remaining default options and click **Next**.

19. In the “Specify the Name and Location” window, enter the following information:
    - Verify that the **Module** field is set to: `FoundationServices`
    - In the **Folder** field, type: `jca`
    - In the **Name** field, change the name of the import component to: `FlatFileOutboundImport`



20. Accept the remaining default options and click **Finish**.
    Wait for the workspace to build.

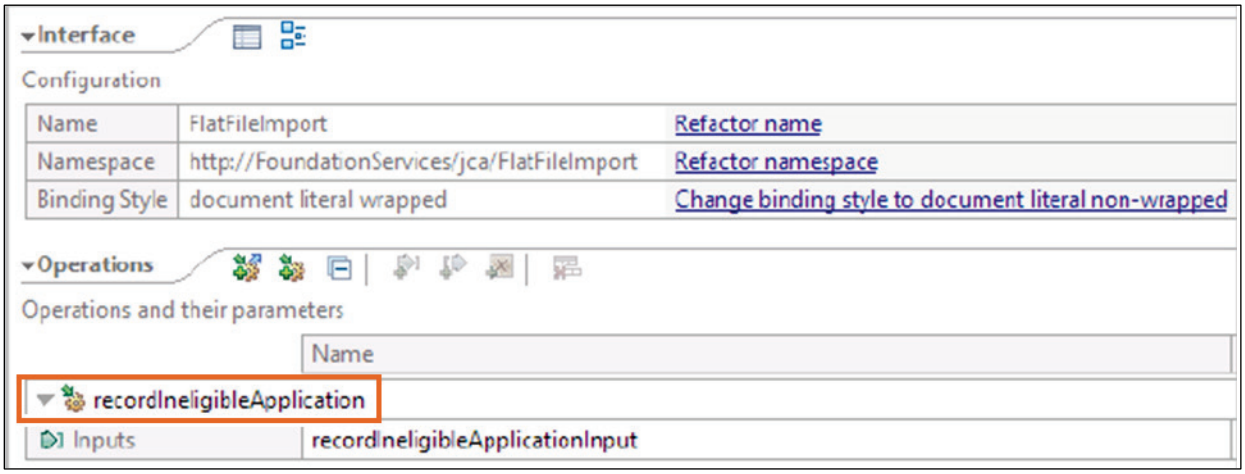
Part 3. Incorporate adapter-related SCA artifacts into an assembly diagram.

1. To examine the artifacts that the External Service wizard generates:

1. In the Business Integration view, expand **FoundationServices** and double-click **Assembly Diagram.**
2. Verify that the **FlatFileOutboundImport** component was added to the assembly diagram of the **FoundationServices** module.

3. In the Business Integration view, expand **FoundationServices > Interfaces > jca** and double-click **FlatFileOutboundImport**.

   The interface has a one-way operation that is named `recordIneligibleApplication`. The operation is one-way because you only chose outbound for the processing direction.

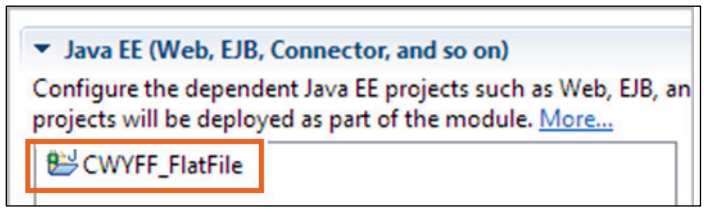


4. Close the interface editor.
5. In the Business Integration view, expand the **CWYFF_FlatFile** project.

Note the contents of the adapter project. In particular, note the `CWYFF_FlatFile.jar` file that contains the FlatFile adapter code and the `CWYBS_AdapterFoundation.jar` file that contains the adapter foundation classes.

6. In the Business Integration view, expand FoundationServices and double-click **Dependencies**.

   The adapter project (`CWYFF_FlatFile`) is added to the Java EE dependencies list.



7. Close the dependency editor.

Part 4. Test an adapter in the IBM Integration Designer test environment.

In this portion of the exercise, you use the newly created FlatFileOutboundImport component to test the WebSphere Adapter for Flat Files. After confirming the configuration of the adapter, you create the remaining components that are needed to process ineligible applications.

To test the flat file adapter:

1. Test the flat file adapter.

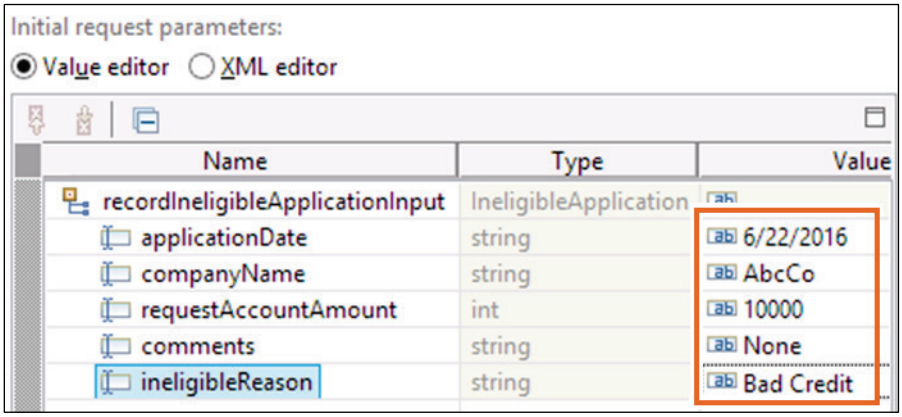
1. Start the process server, if not running, by double-clicking the desktop shortcut that is labeled **Start UTE Process Server**.
2. Right-click **IBM Process Server v8.6 at localhost** and click **Add and Remove** from the menu.
3. Double-click **FoundationServicesApp** in the **Available** list to add the project to the **Configured projects** list.
4. Click **Finish**.

   Wait a few minutes for the application to deploy.
5. If needed, expand the server and check the status of the newly added module. If the **FoundationServicesApp** module has a **Stopped** status in the Servers view, then right-click the module and click **Restart** from the menu. If prompted to do so, republish the module. Continue to the next step when the status changes to **Started**.
6. When the application is deployed and started, the message `Application started: FoundationServicesApp` is displayed in the **Server Logs** view.

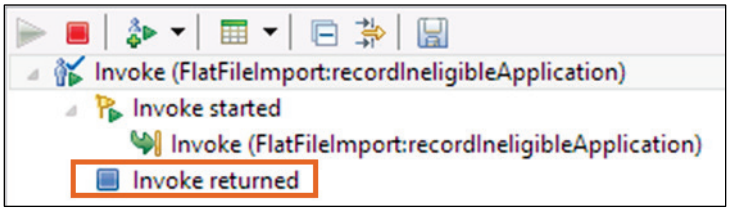
2. Test the FlatFileOutboundImport component on the FoundationServices assembly diagram.

1. In the **FoundationServices** assembly diagram, right-click **FlatFileOutboundImport** and click **Test Component** from the menu.

2. Enter the following test data in the **Initial request parameters** section.
   - applicationDate: `6/22/2016`
   - companyName: `AbcCo`
   - requestAccountAmount: `10000`
   - comments: `None`
   - ineligibleReason: `Bad Credit`

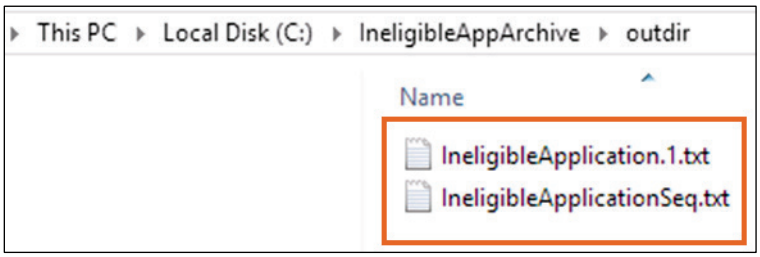


3. Click the **Continue** icon on the Events toolbar to run the test.
4. In the "Select a Deployment Location" dialog box, select **IBM Process Server v8.6 at localhost** and click **Finish**.
5. In the User Login dialog box, accept the default entries for **User ID** and **Password** and click **OK**.
6. The test run is complete when the blue, square **Invoke returned** icon is displayed.



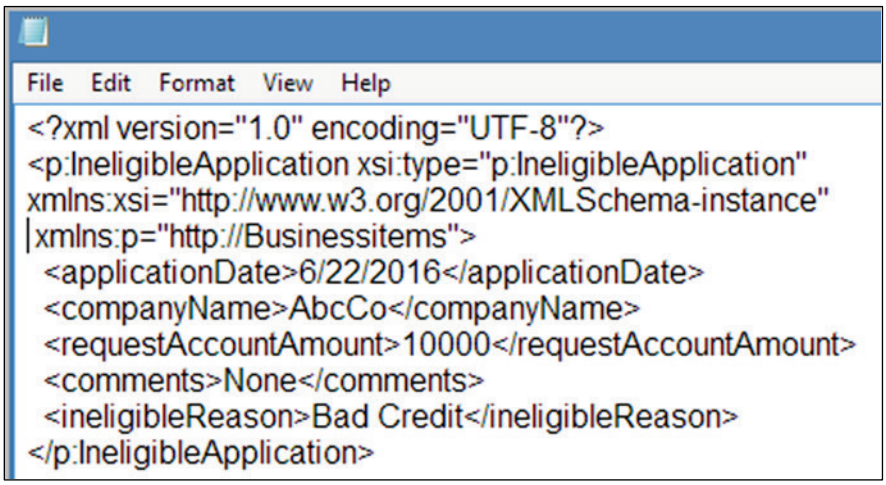
7. Open Windows Explorer and browse to `C:\IneligibleAppArchive\outdir`.

8. If the test was successful, a sequence file that is named `IneligibleApplicationSeq.txt` and an output file that is named `IneligibleApplication.1.txt` are listed.



If you did another test, a file that is named `IneligibleApplication.2.txt` would be generated.

9. Open `IneligibleApplication.1.txt` in a text editor such as Notepad.
10. The file contains the sample data that you entered in the test client.



11. Close `IneligibleApplication.1.txt` and close Windows Explorer.
12. Close the test client tab and click **No** when you are prompted to save the test trace.

2. Remove FoundationServicesApp from the server.

1. In the Servers view, right-click IBM Process Server v8.6 at localhost and click Add and Remove from the menu.
2. Click **Remove All** and click **Finish**. Do not stop the server.

Part 5. Create the RecordIneligibleApplication Java component.

In this portion of the lab, you create a RecordIneligibleApplication Java component. This Java component writes informational messages to the JVM log (`SystemOut.log`) so you can track application progress. The component also creates a Message business object that contains the rejection message: `Account verification recorded this application as ineligible for the customer` <company name>, and it invokes the FlatFileOutboundImport component. The application is then passed to the flat file adapter for serialization to the file system.

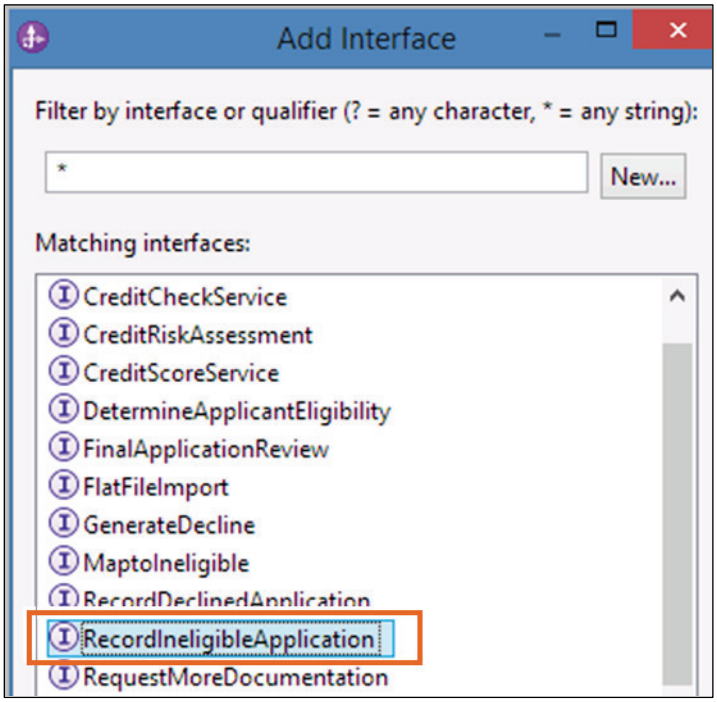
To create the RecordIneligibleApplication component:

1. Add a Java component that is named RecordIneligibleApplication to the FoundationServices assembly diagram.

1. Right-click any blank space on the **FoundationServices** assembly diagram, and click **Add > Java** from the menu.
2. Switch to the **Description** tab in the **Properties** view.
3. Change the **Name** from `Component1` to: `RecordIneligibleApplication`
4. Save your changes.

2. Add the RecordIneligibleApplication interface to the RecordIneligibleApplication Java component.

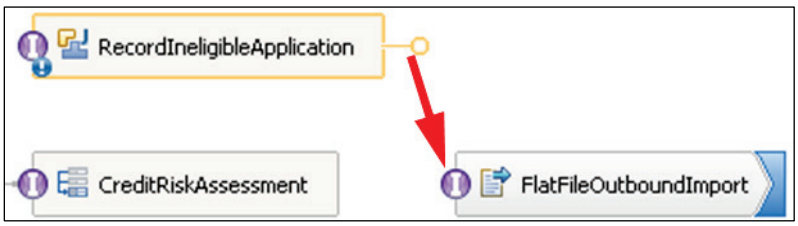
1. Right-click the **RecordIneligibleApplication** Java component, and click **Add > Interface** from the menu.
2. In the Add Interface dialog box, select **RecordIneligibleApplication**.



3. Click **OK**.

3. Wire the RecordIneligibleApplication component to the FlatFileOutboundImport component and create a matching reference on the Java component.

1. Hover the mouse pointer over **RecordIneligibleApplication**.
2. Drag the orange, circular handle of **RecordIneligibleApplication** to the interface of the **FlatFileOutboundImport** component.

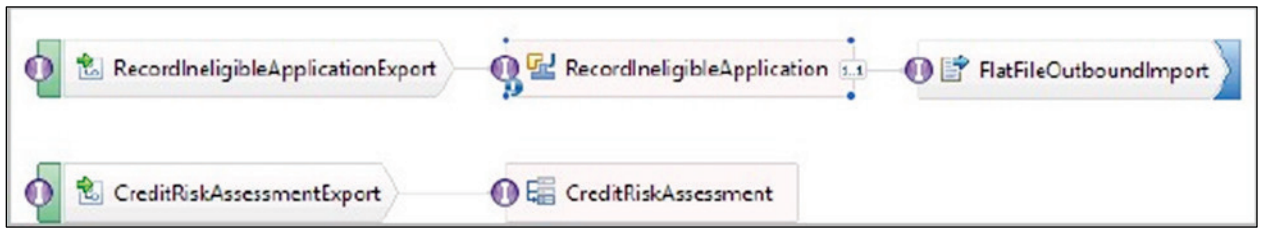


3. Click **OK** in the Add Wire dialog box to create a matching reference on the **RecordIneligibleApplication** component.
4. Save your changes.

4. Generate an export that is named `RecordIneligibleApplicationExport` for the RecordIneligibleApplication Java component. Generate the export with an SCA binding.

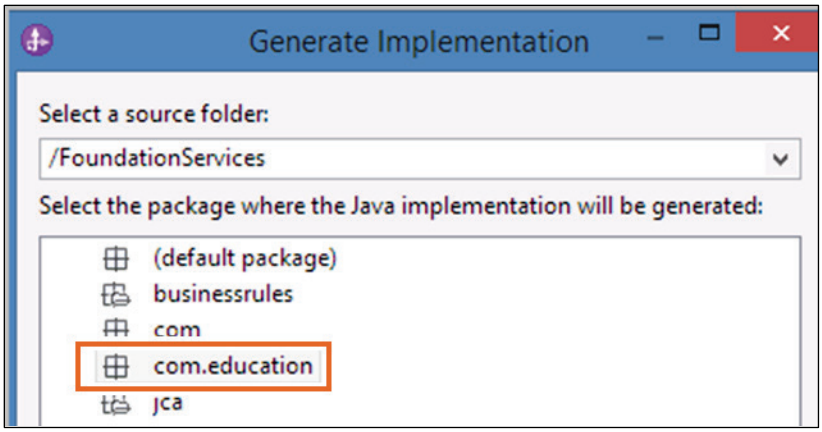
1. Right-click the RecordIneligibleApplication Java component and click Generate Export > SCA Binding from the menu.

2. Accept the default export name: `RecordIneligibleApplicationExport`
3. Save your changes.
   Your assembly diagram resembles the following figure:

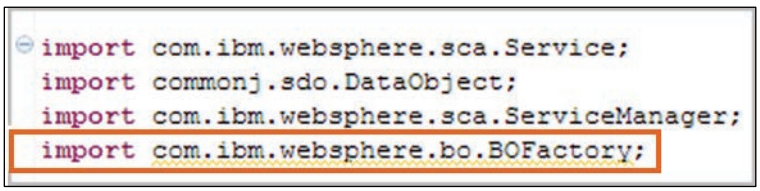


2. Use the snippet code in `C:\labfiles\Support Files\Ex9\JavaMethod_InputCriterion.txt` to generate the implementation for the RecordIneligibleApplication Java component.
   1. Right-click the **RecordIneligibleApplication** Java component and click **Generate Implementation** from the menu.

2. In the Generate Implementation dialog box, take the following actions.
   - Click **New Package**.
   - Type `com.education` in the **Package name** field and click **OK**.
   - When you return to the **Generate Implementation** dialog box, select **com.education**.



3. Click **OK**.
   The RecordIneligibleApplicationImpl.java file opens in the Java editor.
4. At the beginning of the file, click the plus symbol (**+**) to expand the import section.
5. Add the following import statement:
   `import com.ibm.websphere.bo.BOFactory;`



6. Scroll to the `public DataObject InputCriterion` method at the end of the file.
7. In Windows Explorer, browse to `C:\labfiles\Support Files\Ex9`.
8. Open the `JavaMethod_InputCriterion.txt` file in a text editor such as Notepad.

9. Copy the text in `JavaMethod_InputCriterion.txt` and paste it over the green comment lines (that begin with `//`) in the `public DataObject InputCriterion` method. Be sure to remove the final `return null;` from the method.

```
public DataObject InputCriterion(DataObject input) {
    System.out.println("[Java] Record Ineligible Application - begins");
    System.out.println("Account verification failed for customer : "
            + input.getString("companyName"));

    storeFlatFile(input);

    // Construct the Message SDO for return to the calling process
    ServiceManager serviceManager = new ServiceManager();
    BOFactory bof = (BOFactory) serviceManager
            .locateService("com/ibm/websphere/bo/BOFactory");

    DataObject msg = bof.create("http://FoundationLibrary/businessitems", "
    String msgText = "Account Verification recorded this application as ine
            + input.getString("companyName");
    msg.setString("message", msgText);
    System.out.println("[Java] Record Ineligible Application - ends");
    return msg;
}
```

If you see an error marker in the Java editor next to `storeFlatFile(input)`, you can ignore it. The `storeFlatFile` method is implemented shortly.

10. Alternatively, enter the following code:

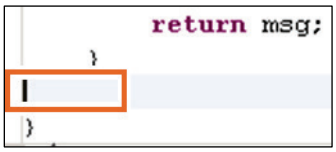
```
System.out.println("[Java] Record Ineligible Application - begins");
System.out.println("Account verification failed for customer : " +
input.getString("companyName"));


storeFlatFile(input);


// Construct the Message SDO for return to the calling process
ServiceManager serviceManager = new ServiceManager(); BOFactory bof
= (BOFactory) serviceManager.
locateService("com/ibm/websphere/bo/BOFactory");


DataObject msg =
bof.create("http://FoundationLibrary/businessitems", "Message");
String msgText = "Account Verification recorded this application as
ineligible for the customer: "
+ input.getString("companyName"); msg.setString("message", msgText);
System.out.println("[Java] Record Ineligible Application - ends");
return msg;
```

11. Close the `JavaMethod_InputCriterion.txt` file but leave Windows Explorer open.
12. Switch to Windows Explorer and open `C:\labfiles\Support Files\Ex9\JavaMethod_storeFlatFile.txt` in a text editor such as Notepad.
13. Copy the text in `JavaMethod_storeFlatFile.txt`.
14. Switch back to the Java editor.
15. Place the cursor just above the last `}` marker in `RecordIneligibleApplicationImpl.java`.



16. Paste the code from `JavaMethod_storeFlatFile.txt` to create a method `storeFlatFile` in `RecordIneligibleApplicationImpl.java`.

```
    public void storeFlatFile(DataObject input) {
        // Set up the Inelig File BO for passing to the Flat File Outbound
        // Interface
        ServiceManager serviceManager = new ServiceManager();
        BOFactory bof = (BOFactory) serviceManager
                .locateService("com/ibm/websphere/bo/BOFactory");

        try {
            // Invoke the Flat File Outbound Interface to append the entry to
            // the file
            System.out.println(" >>> Invoking the Flat File Outbound service ...");
            locateService_FlatFileOutboundImportPartner().invoke(
                    "recordIneligibleApplication", input);
            System.out.println(" <<< Flat File Outbound service invoked OK! ...");
        }
        catch (Throwable t) {
            t.printStackTrace();
        }
    }
}
```

17. Alternatively, enter the following code:

```
public void storeFlatFile(DataObject input) {

ServiceManager serviceManager = new ServiceManager(); BOFactory bof
= (BOFactory) serviceManager
.locateService("com/ibm/websphere/bo/BOFactory");

try {
System.out.println(" >>> Invoking the Flat File Outbound service
..."); locateService_FlatFileOutboundImportPartner().invoke(
"recordIneligibleApplication", input);
System.out.println(" <<< Flat File Outbound service invoked OK!
...");
}
catch (Throwable t) { t.printStackTrace();
}
}
```

18. Press Ctrl+S to save your changes. No error markers are showing in the left margin of the Java editor. If you receive errors, then you can replace the entire text in the implementation with the Java code that is provided in the `RecordIneligibleApplicationImpl.java` file under `C:\Support Files\Ex9`.
19. Close the Java editor.
20. Save the changes to the **FoundationServices** assembly diagram.
21. Close `JavaMethod_storeFlatFile.txt` but leave Windows Explorer open.

Part 6. Test the RecordIneligibleApplication Java component.

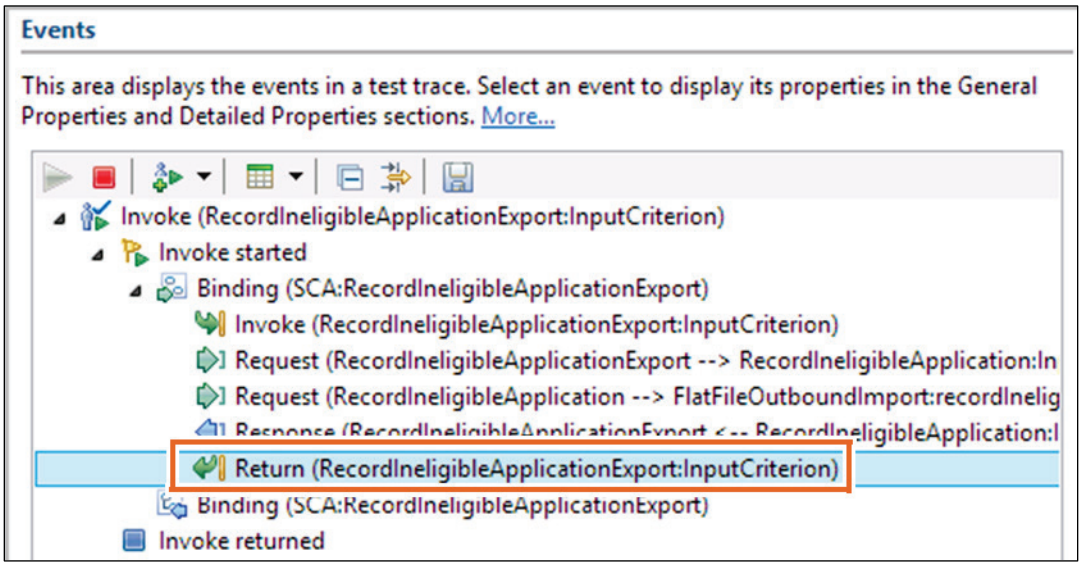
In this portion of the exercise, you test the RecordIneligibleApplicationExport component to verify that both the FlatFileOutboundImport component and the RecordIneligibleApplication component are implemented correctly.

1. Deploy FoundationServicesApp.

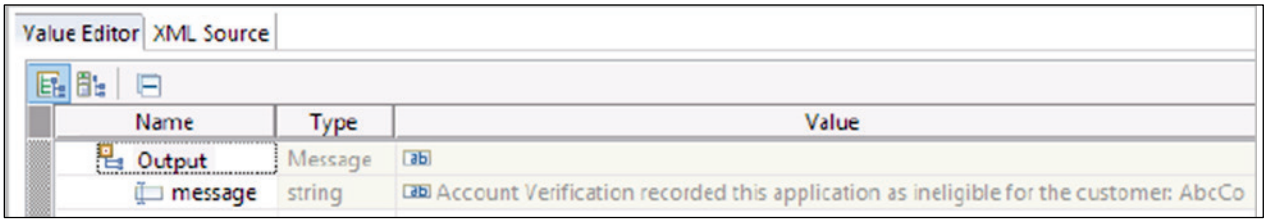
1. Right-click **IBM Process Server v8.6 at localhost** and click **Add and Remove** from the menu.
2. Double-click **FoundationServicesApp** in the **Available** list to add the project to the **Configured projects** list.
3. Click **Finish**.
4. When the application is deployed and started, the message `Application started: FoundationServicesApp` is displayed in the **Server Logs** view.

2. Test the RecordIneligibleApplicationExport component.

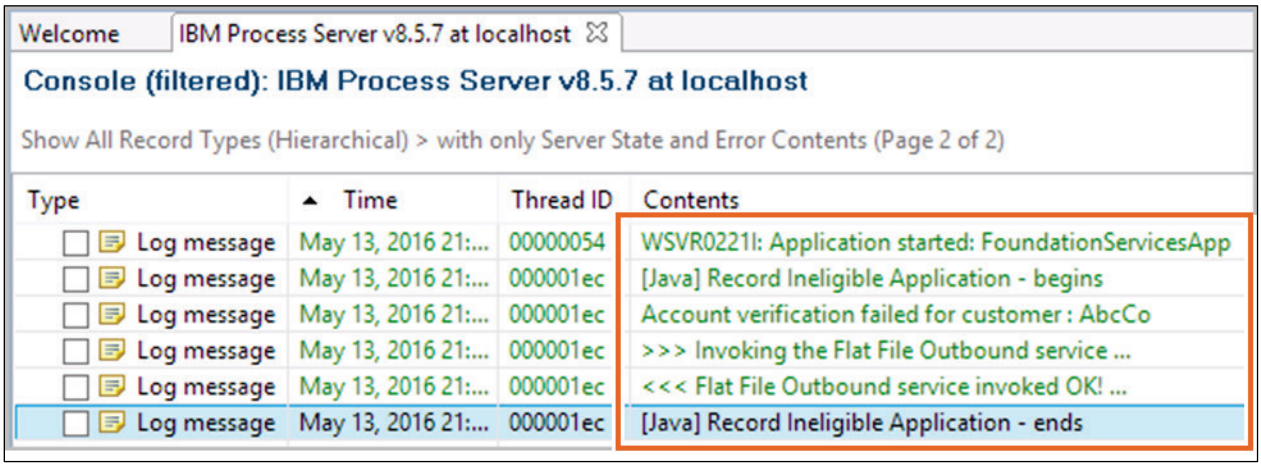
1. On the **FoundationServices** assembly diagram, right-click **RecordIneligibleApplicationExport**, and click **Test Component** from the menu.
2. Enter the following test data in the **Initial request parameters** section.
   - applicationDate: `6/22/2016`
   - companyName: `AbcCo`
   - requestAccountAmount: `10000`
   - comments: `None`
   - ineligibleReason: `Bad credit`
3. Click the **Continue** icon on the Events toolbar to run the test.
4. In the "Select a Deployment Location" dialog box, select **IBM Process Server v8.6 at localhost** and click **Finish**.
5. In the User Login dialog box, accept the default entries for **User ID** and **Password** and click **OK**.
6. When the test completes, select the **Return (RecordIneligibleApplicationExport:InputCriterion)** event.



7. In the **Return parameters** section, the value in the **message** field is: `Account Verification recorded this application as ineligible for the customer: AbcCo`



Messages that the **RecordIneligibleApplication** component returned are displayed in the Server Logs view.



8. In Windows Explorer, browse to `C:\IneligibleAppArchive\outdir`.
9. Verify that a new `IneligibleApplication.txt` file was generated in `C:\IneligibleAppArchive\outdir`.

   The file is named `IneliegibleApplication.2.txt`.
10. Close Windows Explorer.
11. Close the test client tab and click **No** when you are prompted to save the test trace.

2. Remove FoundationServicesApp from the server.

1. In the **Servers** view, right-click **IBM Process Server v8.6** and click **Add and Remove** from the menu.
2. Click **Remove All** and click **Finish**.

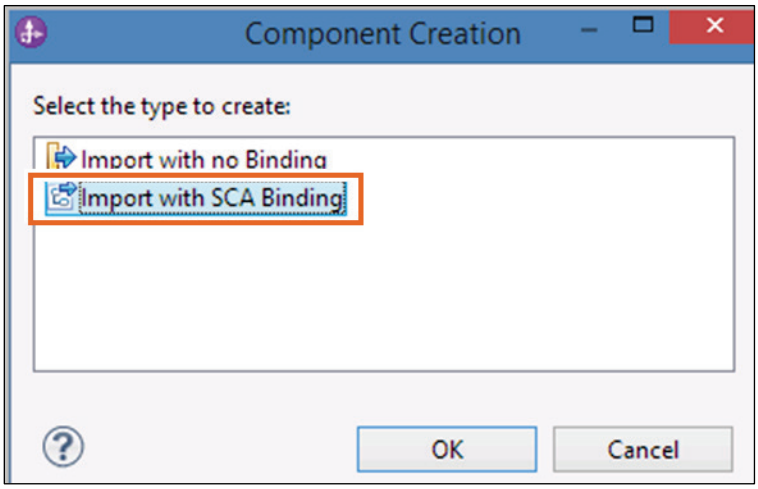
3. Wire the service to the business process

The AccountVerification process in FoundationModule invokes the RecordIneligibleApplication service to archive ineligible applications to the file system (by using the WebSphere Adapter for Flat Files). In this portion of the exercise, you create an import component on the FoundationModule assembly diagram that calls the RecordIneligibleApplicationExport component in FoundationServices. You then wire the import component to the AccountVerification process.

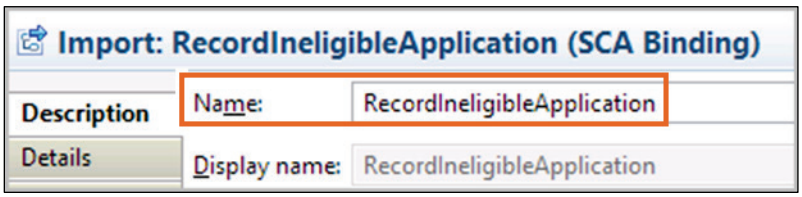
1. Open the **FoundationModule** assembly diagram.
2. In the Business Integration view, expand **FoundationModule**.
3. Double-click **Assembly Diagram**.

4. Create an import component on the FoundationModule assembly diagram, named: `RecordIneligibleApplication`

1. In the Business Integration view, expand **FoundationServices > Assembly Diagram**.
2. Drag **RecordIneligibleApplicationExport** onto the **FoundationModule** assembly diagram.
3. In the Component Creation dialog box, select **Import with SCA Binding**.



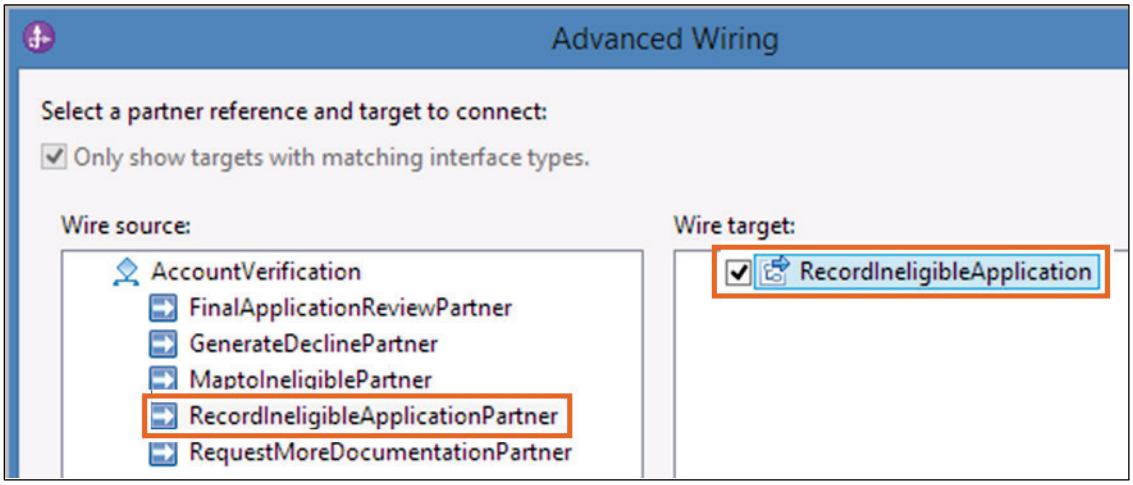
4. Click **OK**.
5. Switch to the **Description** tab in the **Properties** view.
6. Change the **Name** of the import component to: `RecordIneligibleApplication`



7. Save your changes.

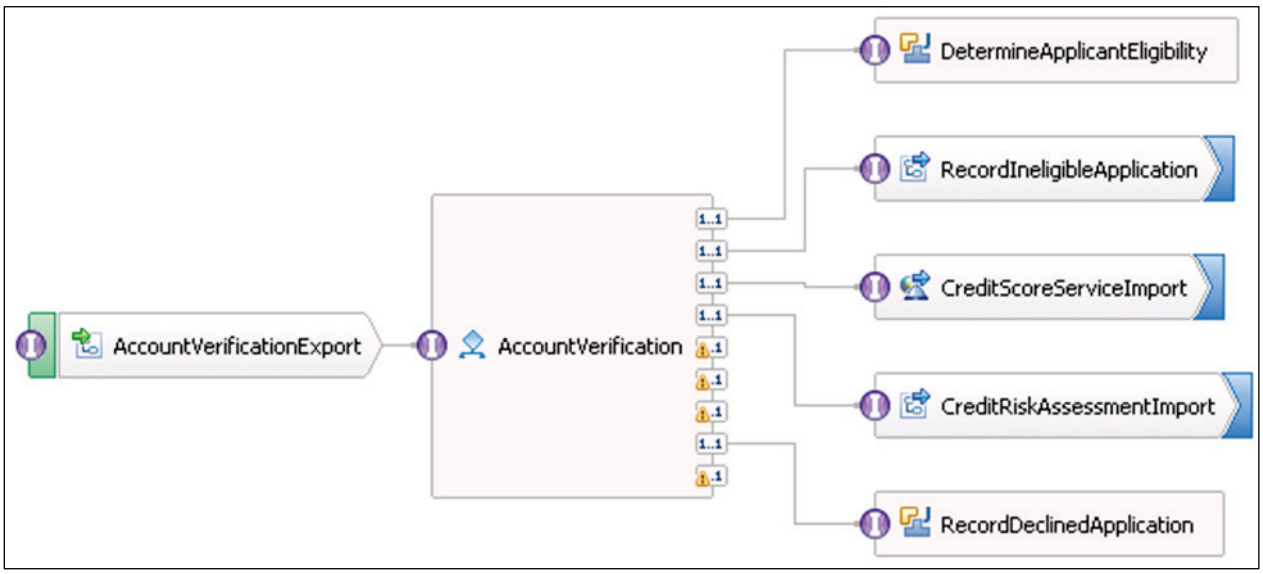
3. Wire the RecordIneligibleApplicationPartner reference of the AccountVerification process to the RecordIneligibleApplication import.

1. Right-click the **AccountVerification** process component and click **Wire (Advanced)** from the menu.
2. In the Advanced Wiring dialog box, in the “Wire source” window, select **RecordIneligibleApplicationPartner.**
3. In the “Wire target” window, select **RecordIneligibleApplication**.



4. Click **OK**.
5. Save your changes.

   The **FoundationModule** assembly diagram resembles the following figure:



6. Close IBM Integration Designer.

Results:
In this exercise, you configured the WebSphere Adapter for Flat Files and tested the adapter in the IBM Integration test environment.

IBM Training

References

- WebSphere Business Integration Adapter Information Center:
  - http://www.ibm.com/software/integration/wbiadapters/library/infocenter/
- Integrate WebSphere Business Integration Adapters with WebSphere Process Server Version 6: Application Event Notification (AgentDelivery) scenario:
  - http://www.ibm.com/developerworks/websphere/library/techarticles/0601_reddy/0601_reddy.html
- Java Connector Architecture:
  - http://java.sun.com/j2ee/connector/index.jsp
- IBM Redbooks for WebSphere Adapter Development:
  - http://www.redbooks.ibm.com/abstracts/sg246387.html
- WebSphere Adapters home page:
  - http://www.ibm.com/software/integration/wbiadapters

References

IBM Training

Developing mediation services

IBM Business Process Manager V8.6

IBM Training

Unit objectives

- Describe the role of mediation services in IBM Process Server
- Define the concept of mediation modules
- Describe how to create mediation flows in IBM Integration Designer
- Describe the role of SMOs in mediations
- Explain the structure of SMOs

Unit objectives

IBM Training

Topics

- Mediation services
- Service message objects

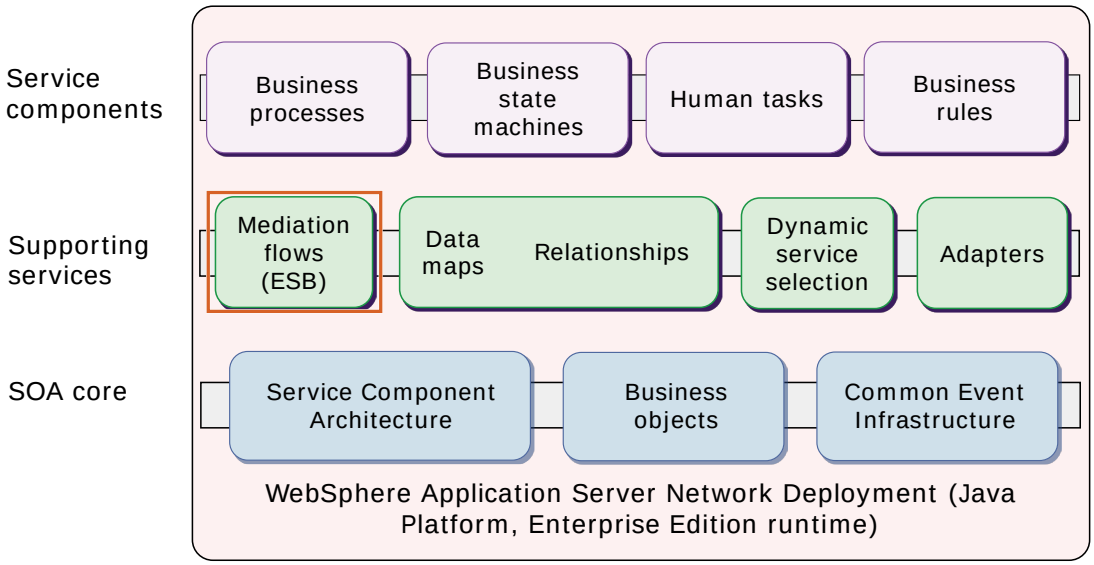
Topics

Mediation services

IBM Training

Mediation flows are supporting services

- Mediation modules can be deployed to IBM Process Server
- No equivalent support in IBM Process Designer



Developing mediation services

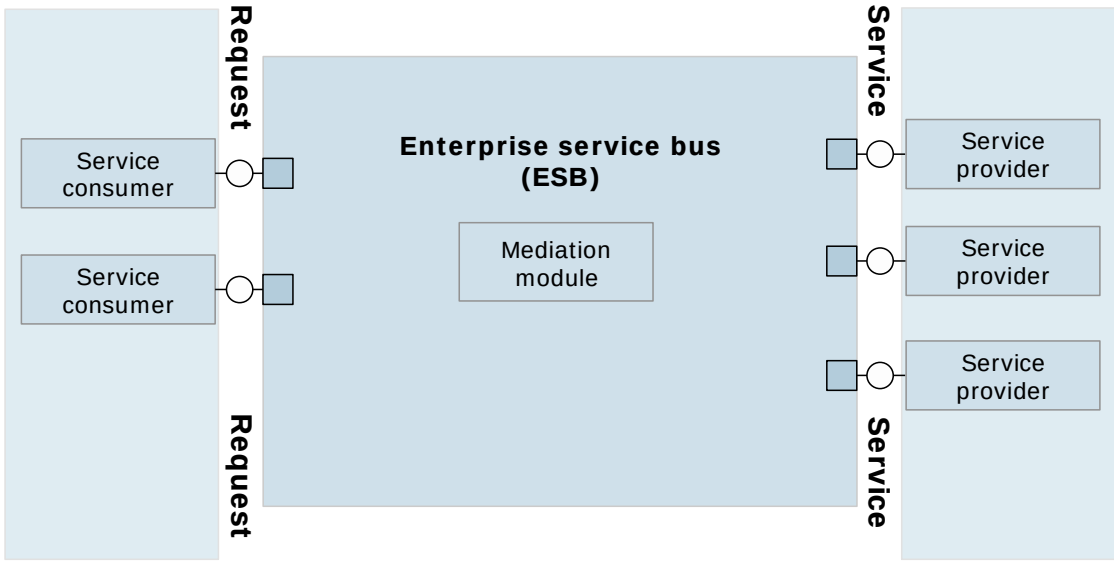
Mediation flows are supporting services

Above the SOA core, IBM built a set of added services that provide higher-level abstractions and ease-of-use capabilities. These functions are mediation flows, data maps, relationships, dynamic service selection components (selectors, mediation primitives), and adapters. Except for selectors, these components can be used in both IBM Process Server business integration modules and mediation modules.

IBM Training

What does an enterprise service bus do? (1 of 2)

- An ESB supplies a communication layer to support service interactions
  - It should support communication through various protocols



Developing mediation services

What does an enterprise service bus do?

In the SOA realm, services do not communicate directly with each other. The enterprise service bus receives the messages from the service requester; it does the required routing, converting, and transforming of the message, and then passes it to the service provider.

IBM Business Process Manager Advanced provides one of the keys to helping you achieve the goals of SOA. It provides a flexible connectivity infrastructure for integrating applications and services, enabling composite applications to be built as a loose coupling of independent services. It reduces the number, size, and complexity of interfaces and connections that must be defined and maintained.

IBM Training

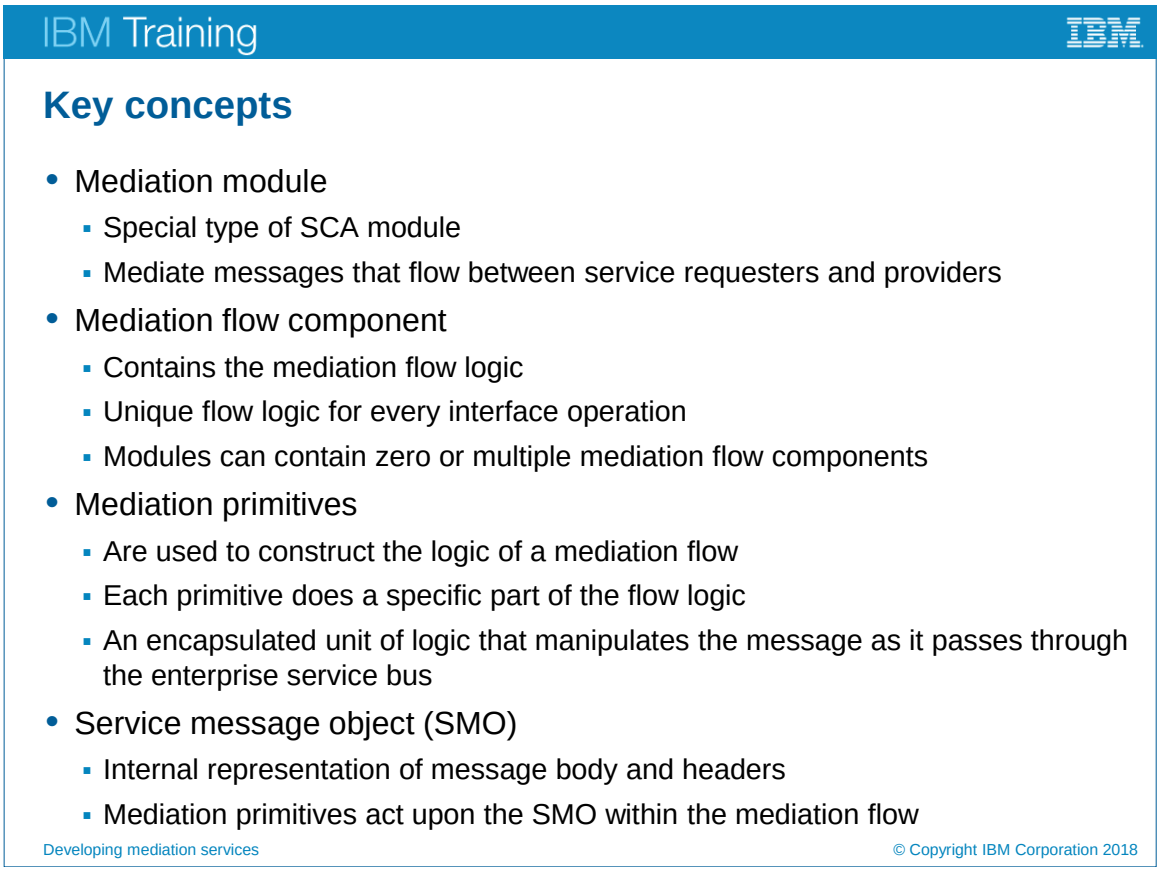
What does an enterprise service bus do? (2 of 2)

- ESB reduces the tight coupling between the service consumer and provider
- The service consumer is ***not*** affected if:
  - The service provider location changes
  - The service provider interface changes
  - A different service provider is used
- An ESB does the following actions between the service consumer and service provider:
  - **Route** messages between services
  - **Convert** transport protocols between consumer and provider
  - **Transform** message formats between consumer and provider
  - **Handle** business events from disparate sources

Developing mediation services

An enterprise service bus provides four primary functions:

- **Routing** messages: The requester sends the request to the ESB, and the ESB is responsible for calling the appropriate service provider.
- **Converting** transport protocols: The ESB allows the service requester to use one transport protocol while the service provider uses another.
- **Transforming** message formats: ESB eliminates the direct call from the service requester to the service provider. In this way, the ESB can modify the message, so the interfaces that the requester and provider use do not have to be identical.
- **Handling** business events: Events can be handled from disparate sources.

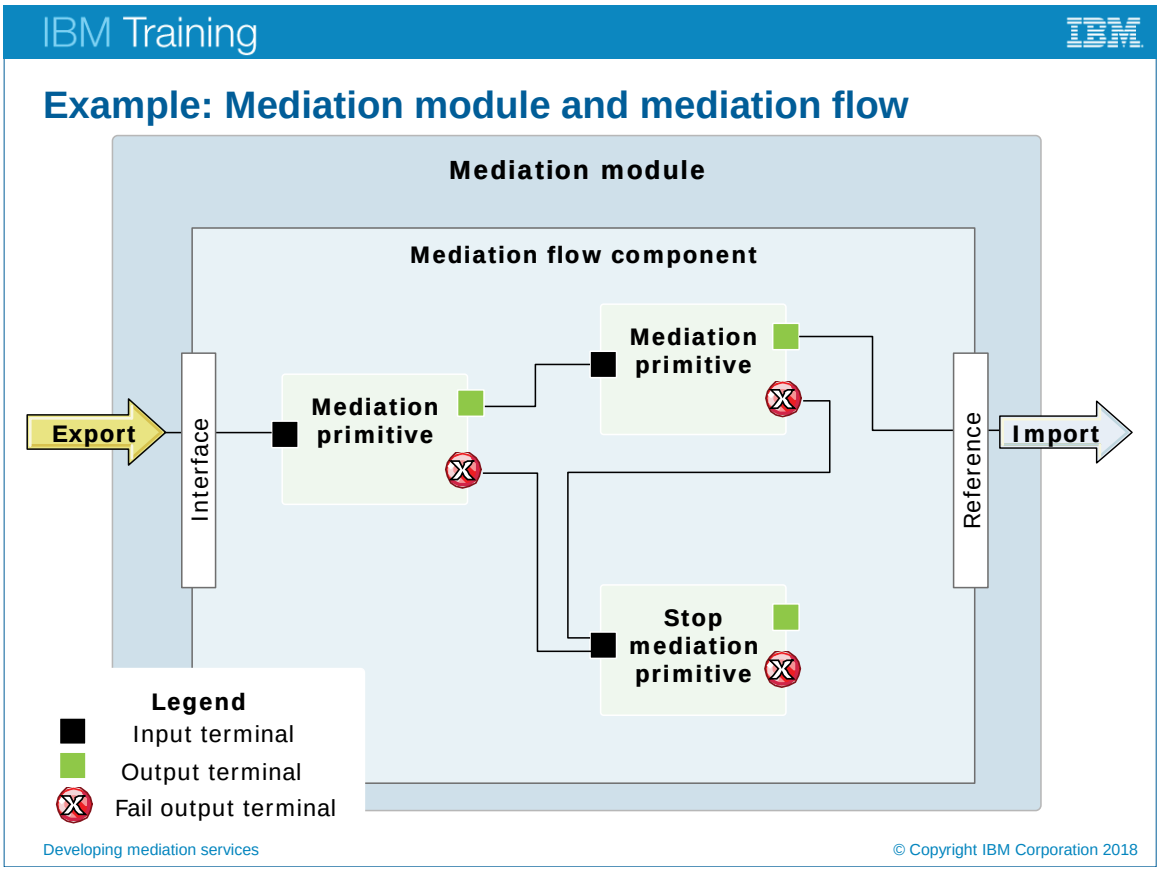


Key concepts

Mediation modules are a special type of SCA modules that can change the format, content, or target of service requests. They use SCA exports and imports to communicate with service requesters and service providers, which provide the key to handling protocol conversions within the bus.

The mediation module also contains a mediation flow component. The mediation flow component contains the logic for the mediation. For every operation defined on an input interface, a unique mediation flow logic is defined for the request and response of the operation. The mediation flow logic transforms and dynamically routes messages.

The flow logic is defined in mediation flow components by using mediation primitives. Each mediation primitive provides some specific portion of the logic and is wired to other primitives into a logical flow. A service message object represents the data that mediation primitives manipulate.



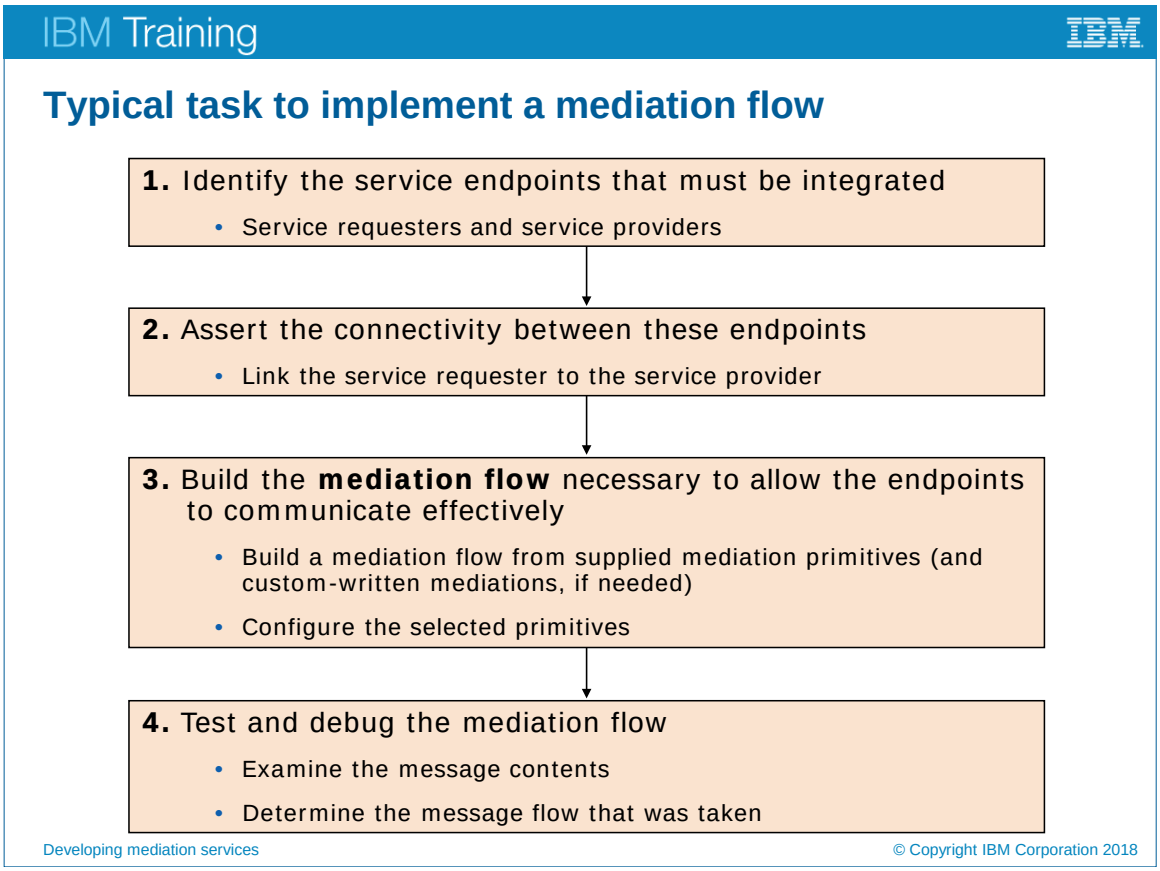
Example: Mediation module and mediation flow

The diagram shows a simplified example of a mediation module. The mediation module contains one mediation flow component, which contains mediation primitives.

Usually, mediation modules contain a specific type of SCA component that is called a mediation flow component. Mediation flow components define mediation flows.

A mediation flow component can contain none, one, or a number of mediation primitives. IBM Business Process Manager supports a supplied set of mediation primitives that provide functions for message routing and transformation. For more mediation primitive flexibility, use the Custom Mediation primitive to call custom logic.

The purpose of a mediation module that does not contain a mediation flow component is to transform service requests from one protocol to another. For example, a service request might be made by using SOAP/JMS but might need transforming to SOAP/HTTP before sending on. You can view and make certain changes to mediation modules from IBM Business Process Manager. However, you cannot view or change the SCA components inside a module from IBM Business Process Manager. Use Integration Designer to customize SCA components.



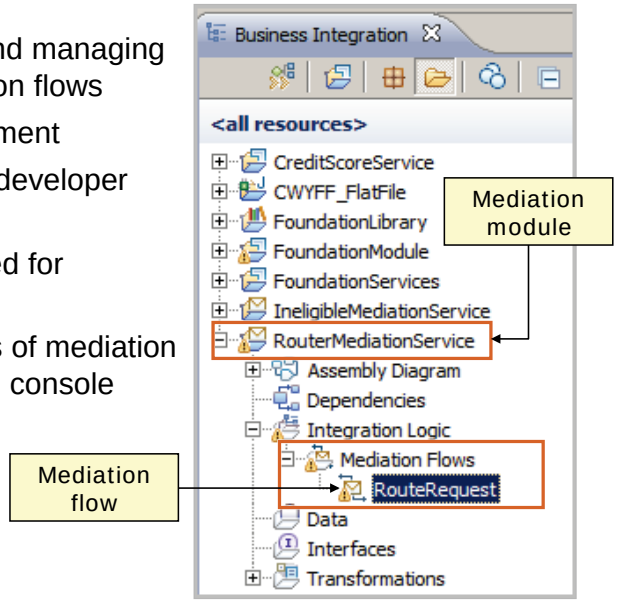
Typical task to implement a mediation flow

This visual describes the typical tasks that an integration designer does when developing an integration solution by using a mediation flow.

IBM Training

Mediation module and mediation flow: Tools (1 of 2)

- IBM Integration Designer tools:
  - Easy-to-use tools for defining and managing mediation modules and mediation flows
  - Eclipse-based tools for development
  - Focused on the more technical developer role
  - Business Integration view is used for interaction with mediation flow
  - You can change certain aspects of mediation modules from the administrative console without having to redeploy the module



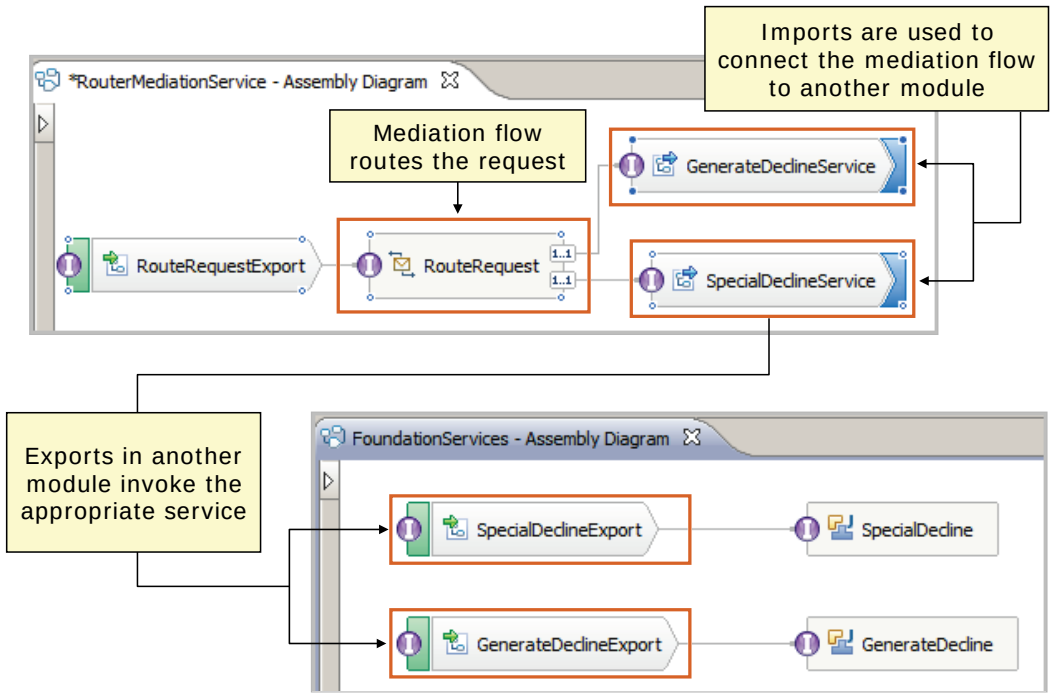
Developing mediation services

Mediation module and mediation flow: Tools

For mediation primitive properties to be visible from the IBM Business Process Manager administrative console, the integration developer must promote the properties. Certain properties lend themselves to being administratively configured, and Integration Designer describes these properties as promotable properties because they can be promoted from the integration cycle to the administrative cycle. Other properties are not suitable for administrative configuration because modifying them can affect the mediation flow in such a way that the mediation module needs to be redeployed. Integration Designer lists the properties that you can choose to promote under the promoted properties of a mediation primitive.

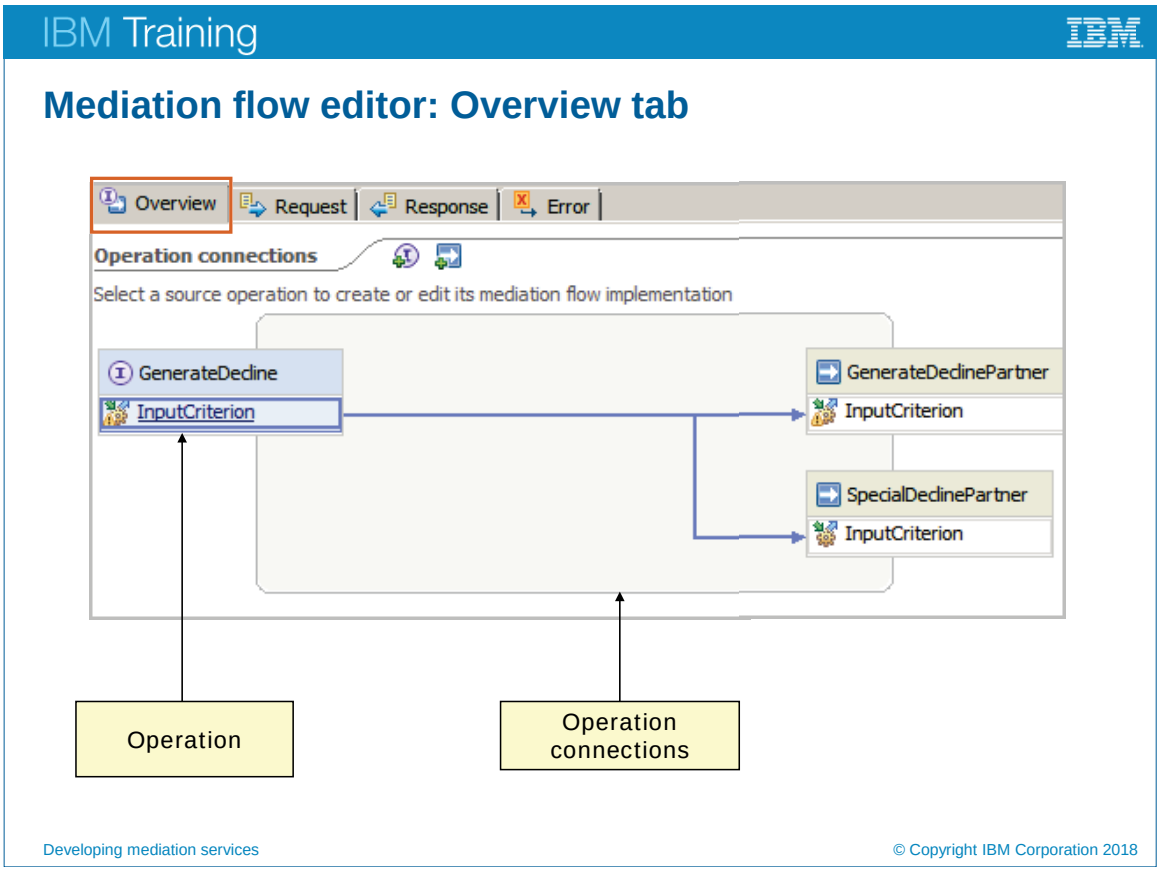
IBM Training

Mediation module and mediation flow: Tools (2 of 2)



Developing mediation services

Assembly editor is used to connect a mediation flow component to calling SCA components.



Mediation flow editor: Overview tab

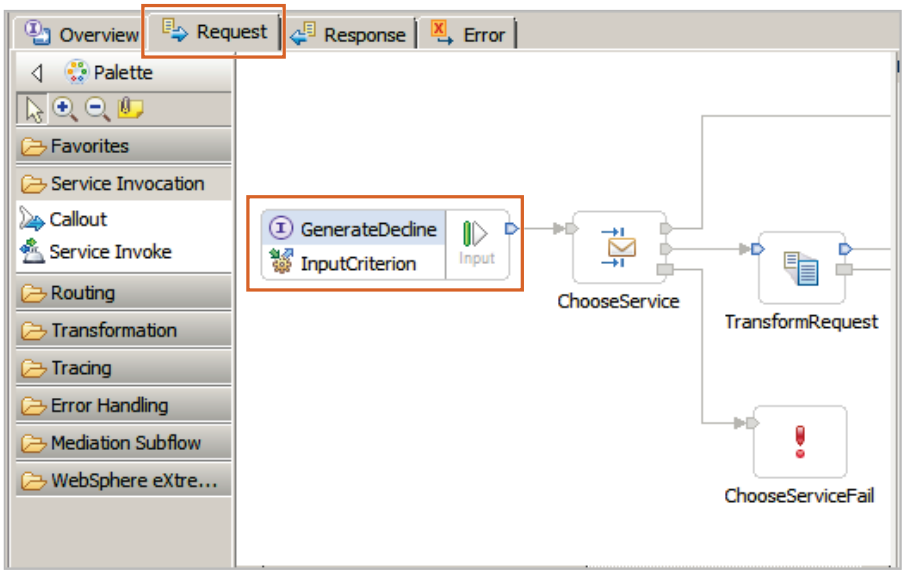
Mediation flow editor has two main sections:

- Operation connections section (the **Overview** tab): Mappings between the corresponding operations of interfaces and references are displayed.
- Request mediation flow: If the mediation is a two-way operation, the request and response flows are shown.

IBM Training

Mediation flow editor: Request flow view

- Mediation primitives from the palette are wired together in the drawing canvas: Request view



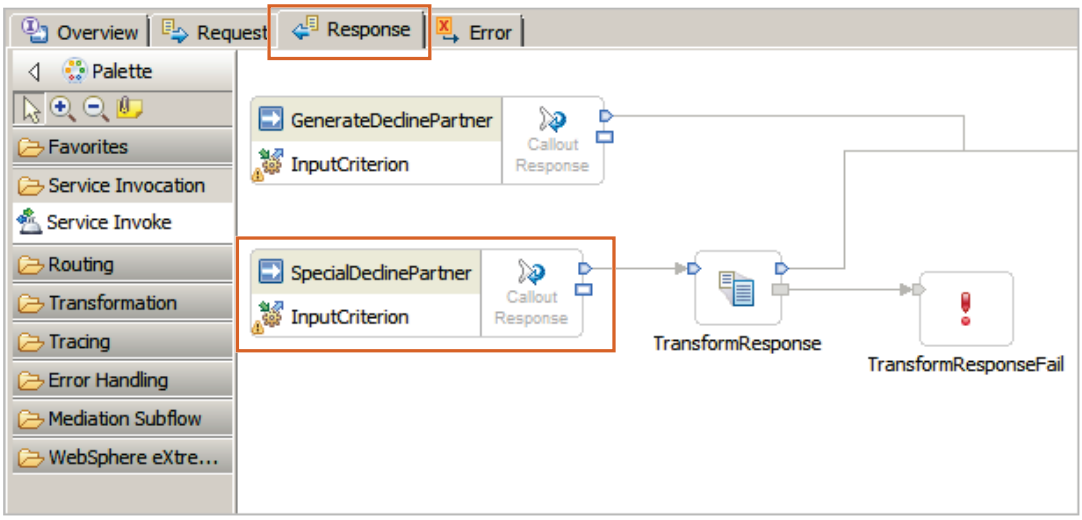
Developing mediation services

Mediation flow editor: Request flow view

IBM Training

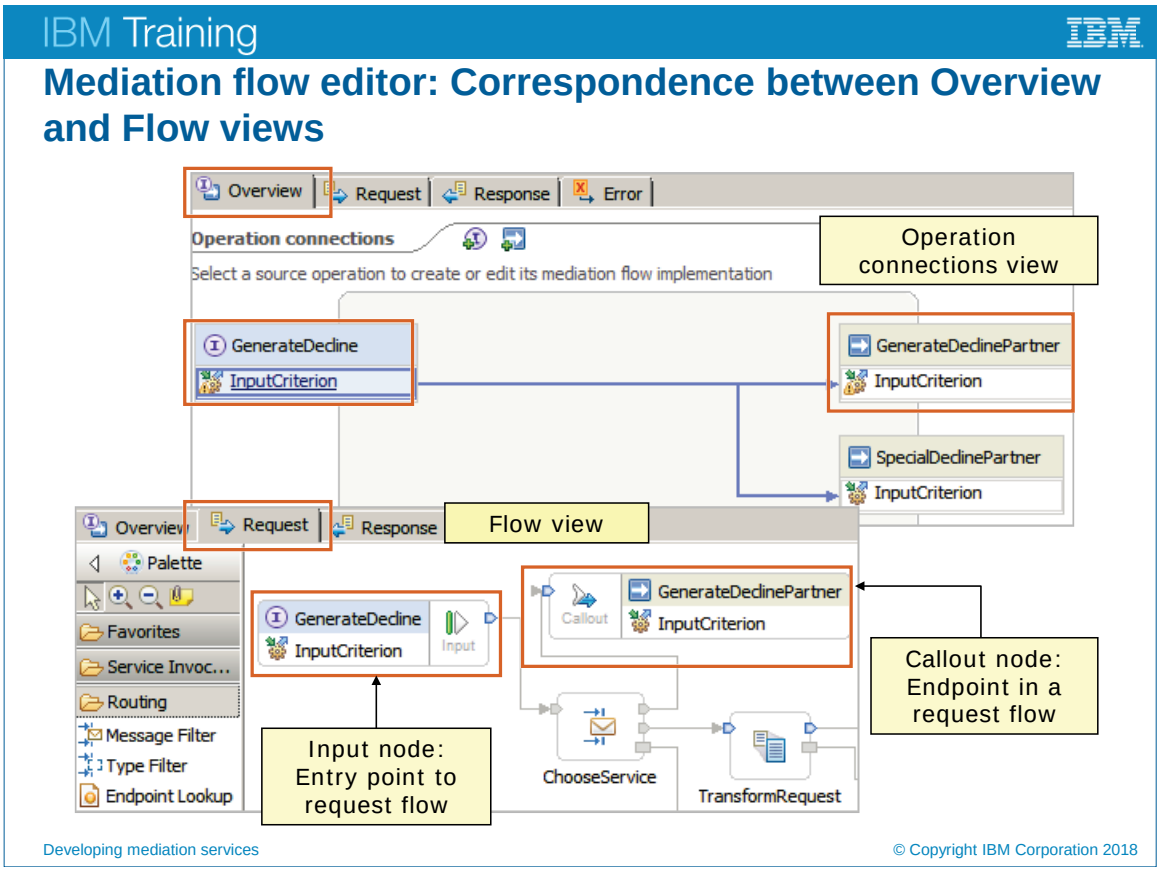
Mediation flow editor: Response flow view

- Mediation primitives from the palette are wired together in the drawing canvas: Response view



Developing mediation services

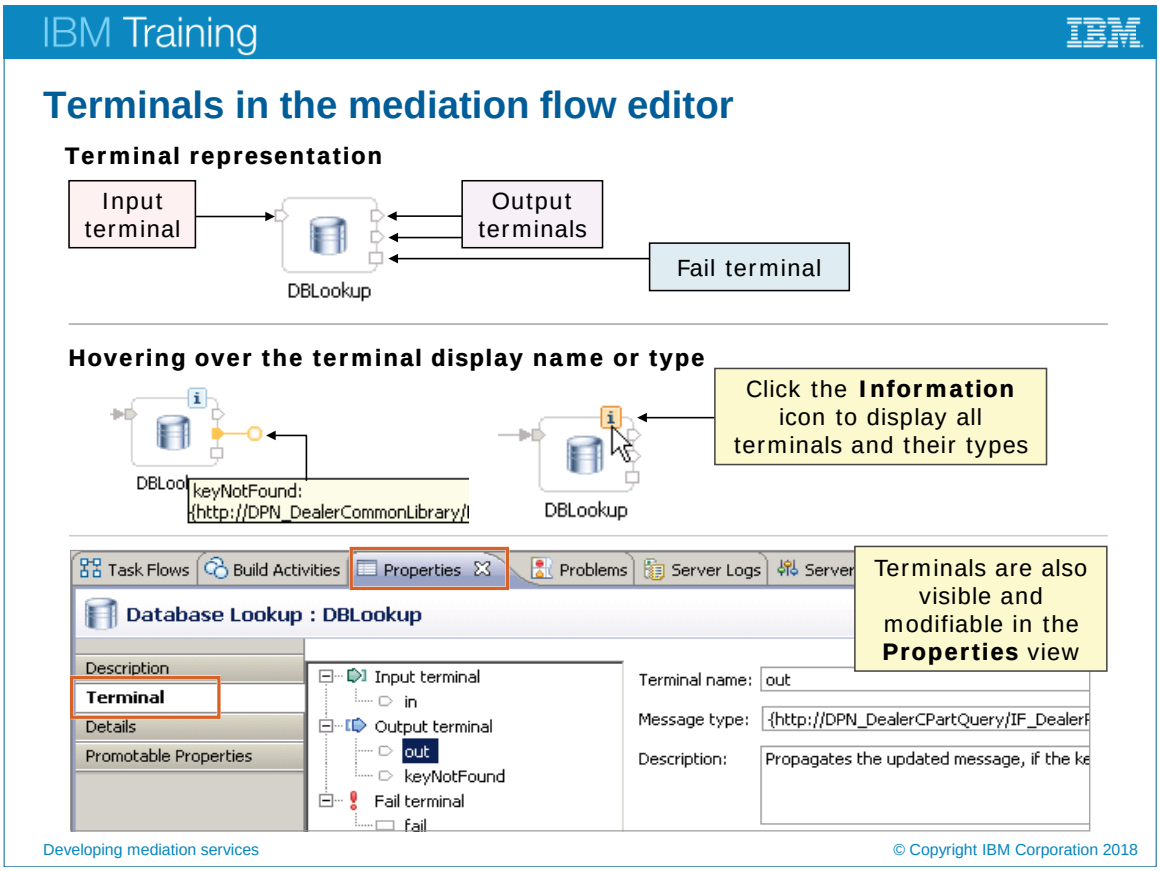
Mediation flow editor: Response flow view



Mediation flow editor: Correspondence between Overview and Flow views

The operation connection in the Overview perspective is mapped from the **service requester** (or export) to the **service provider** (or **import**). The Operation connections view shows connections to GenerateDeclinePartner and SpecialDeclinePartner.

The Flow view shows the detailed functional steps that take place across this connection after it enters the input node of GenerateDecline.



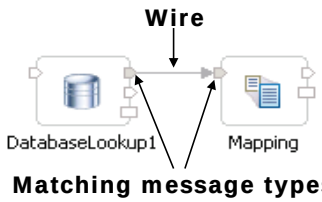
Terminals in the mediation flow editor

Terminals have a message type that is associated with them. These types are based on the WSDL message types.

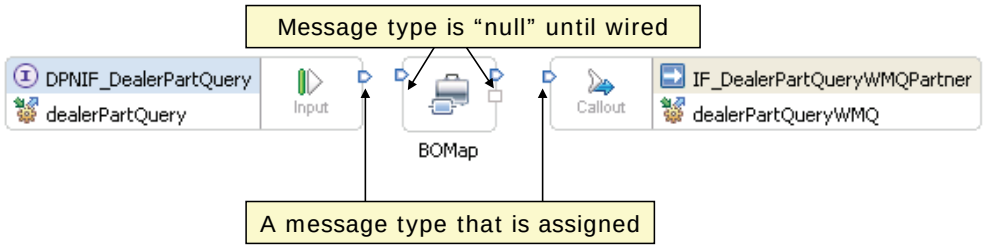
IBM Training

Wiring of mediation primitive terminals

- Connections between terminals are represented as wires
- A connection must have matching terminal message types



- The editor dynamically manages the terminal message types



Developing mediation services

Wiring of mediation primitive terminals

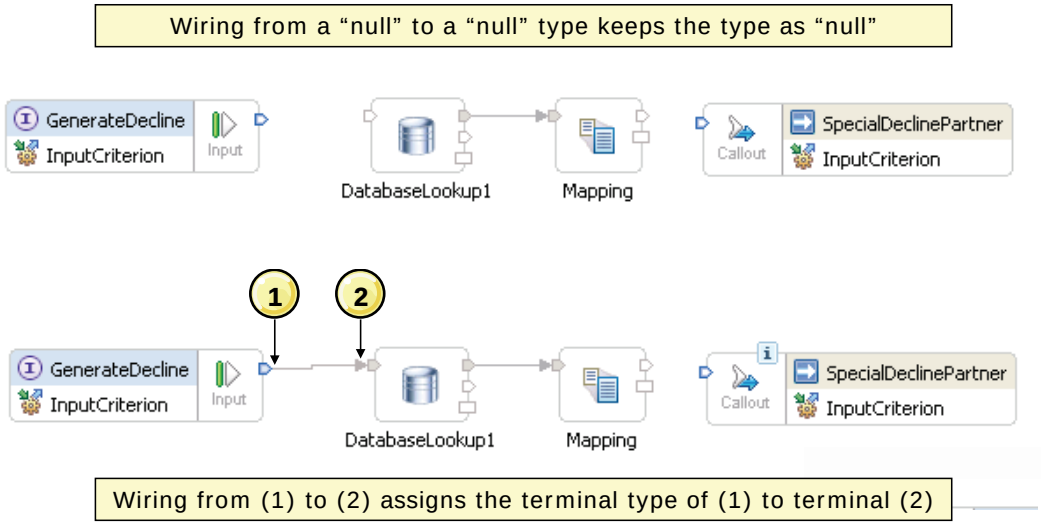
The following notes apply to the wiring of primitive terminals:

- Connections between terminals are represented as wires.
- A connection must have matching terminal message types.
- The editor dynamically manages terminal message types.
  - Input and callout nodes have terminal message types that are fixed.
  - Primitives have terminal message types that are dynamically configured.

IBM Training

Example of wiring of mediation primitive terminals

- Example of wiring and message type

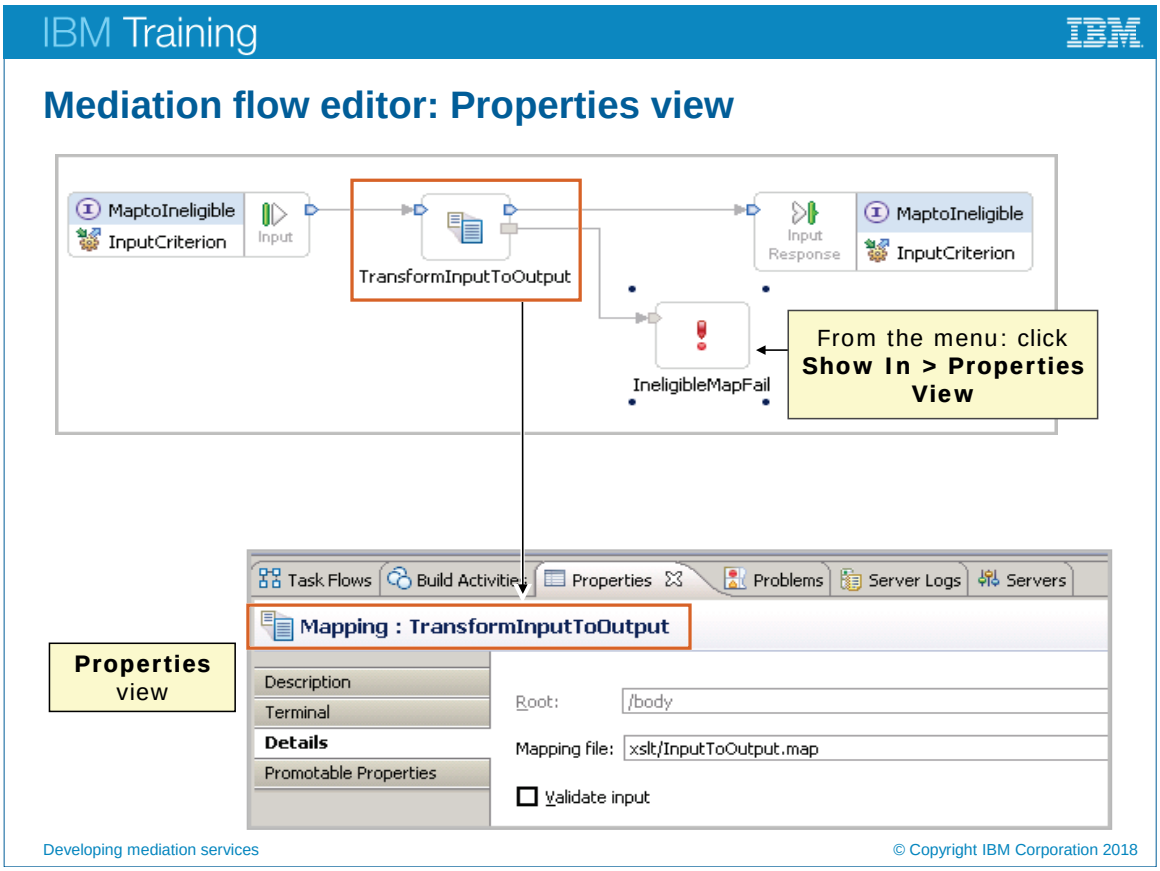


Developing mediation services

Example of wiring of mediation primitive terminals

To wire two primitives together, the type of the message that flows from one terminal to the other must match, or the terminal must accept a message of “anyType” (with some exceptions).

If you attempt to wire terminals that do not have the same message type, IBM Integration Designer warns you. It provides options to help you resolve the type mismatch, such as inserting a primitive to convert the message type.



Mediation flow editor: Properties view

The properties of a primitive control how the primitive operates.

You can view the properties for any artifact in the mediation flow, such as a primitive or line. Switch to the Properties perspective and then select the artifact, or right-click the artifact and click **Show In > Properties View** from the menu. You use this view frequently when you are developing a mediation flow.

IBM Training

Mediations subflow: Overview

- Facilitates the reuse of mediation logic without rewriting
- Composed of Mediation primitives that are wired together
  - Start with an in primitive and end with an out primitive
  - The primitives act like input and callout nodes of a mediation flow
- Used by a parent as if it were a primitive:
  - After a subflow is developed, it is dropped onto the canvas from the primitive as if it were a built-in primitive
  - Wired into the calling flow like any other primitive
- The caller can set the promoted properties of a subflow
- Subflows can be nested
- Can be contained in a library or module (a library increases reusability)

Developing mediation services

Mediations subflow: Overview

A subflow is a preconfigured set of mediation primitives that you wire together to implement a common function. A mediation subflow is run in the context of a parent flow, and can be reused in mediation flows or in subflows.

A mediation subflow can be **created** in a module, mediation module, or library. A mediation subflow can be **used** only within a mediation flow. It is a good practice to store subflows in libraries so that they can be easily shared between mediation flows in different modules.

A mediation subflow has one or more in nodes, followed by one or more mediation primitives, followed by one or more out nodes. Unlike mediation flows, the inputs and outputs of a subflow are not tied to specific interfaces. Instead, the in and out nodes represent the start and end points of a subflow. In a wired mediation flow, these nodes have message types that are associated with them. The message types on these nodes are not defined when the subflow is created, and they can be changed later.

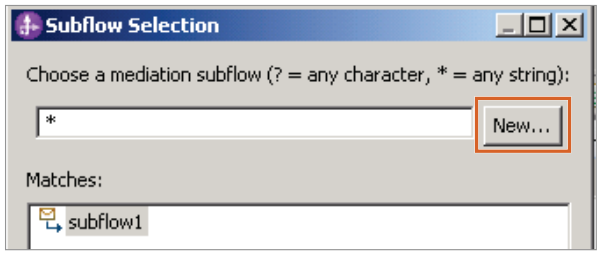
You define a subflow implementation by using the subflow editor. When it is defined, it is displayed as a new type of primitive that you embed in the parent flow, at which point it is expanded into the parent flow. From that perspective, it is more like a macro than an externally callable function.

Because subflows run in the context of a parent flow, properties that are promoted in a subflow are propagated up to the mediation flow in which the subflow is used. If you define a mediation flow that invokes a subflow, the promoted properties for the subflow are visible with the promoted properties of the parent flow.

IBM Training

Mediations subflow: Implementation

- To create a subflow, select the subflow primitive from the palette and place it on the canvas
- The Subflow Selection dialog box is displayed so you can choose an existing subflow or create one
- When you are creating a subflow, the subflow editor opens and places in and out nodes on the canvas
- You add the remaining nodes, and then configure and wire them



Developing mediation services

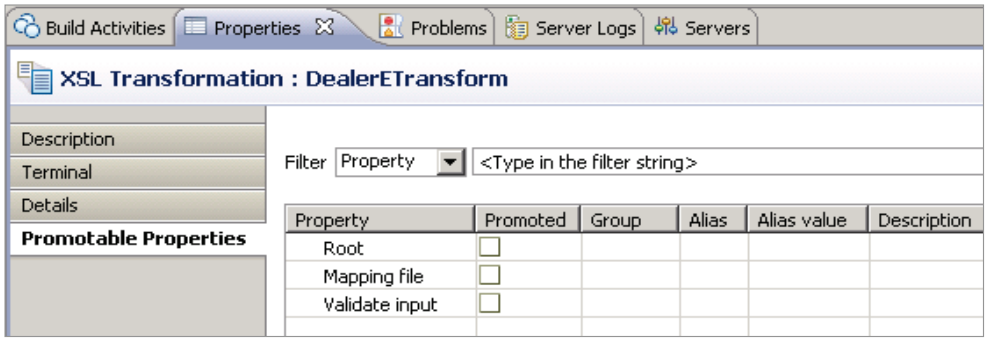
Mediations subflow: Implementation

The subflow editor provides the starting nodes (in and out), and then you code whatever primitives are needed to implement the required logic. When you save the subflow, it is available for reuse whenever you select the subflow primitive from the palette.

IBM Training

Promoted properties: Overview

- “Promoting” the properties of a mediation primitive allows them to be changed at run time



- Properties are changed at run time by using the administrative console; no server restart or redeployment of the mediation is required
- Not all primitive properties can be promoted

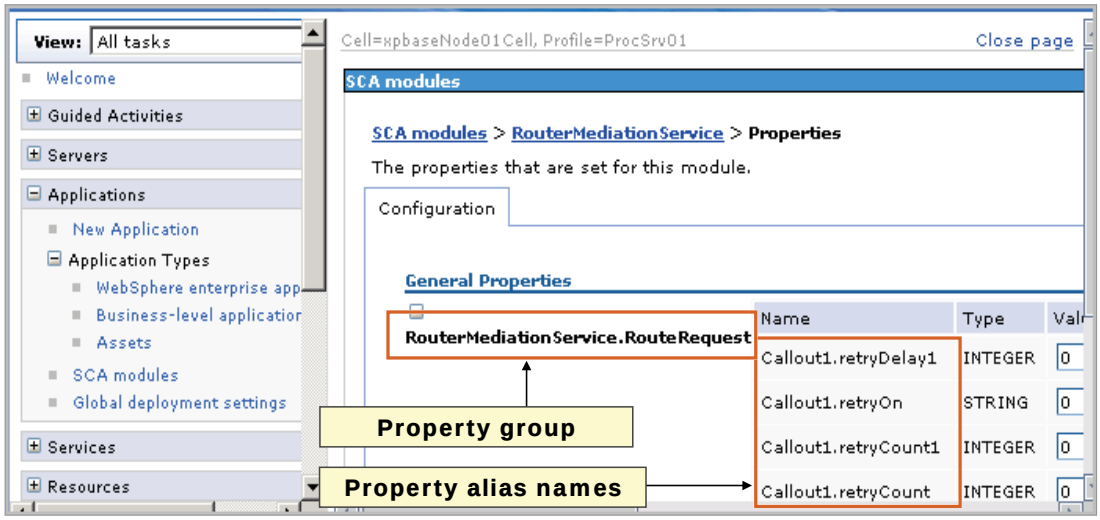
Developing mediation services

Promoted properties: Overview

By using promoted properties, you can add flexibility by being able to change how primitives function without having to change them in IBM Integration Designer.

IBM Training

Promoted properties: Administrative management



Administrator manages promoted properties from the Integrated Solutions Console

Developing mediation services

Promoted properties: Administrative management

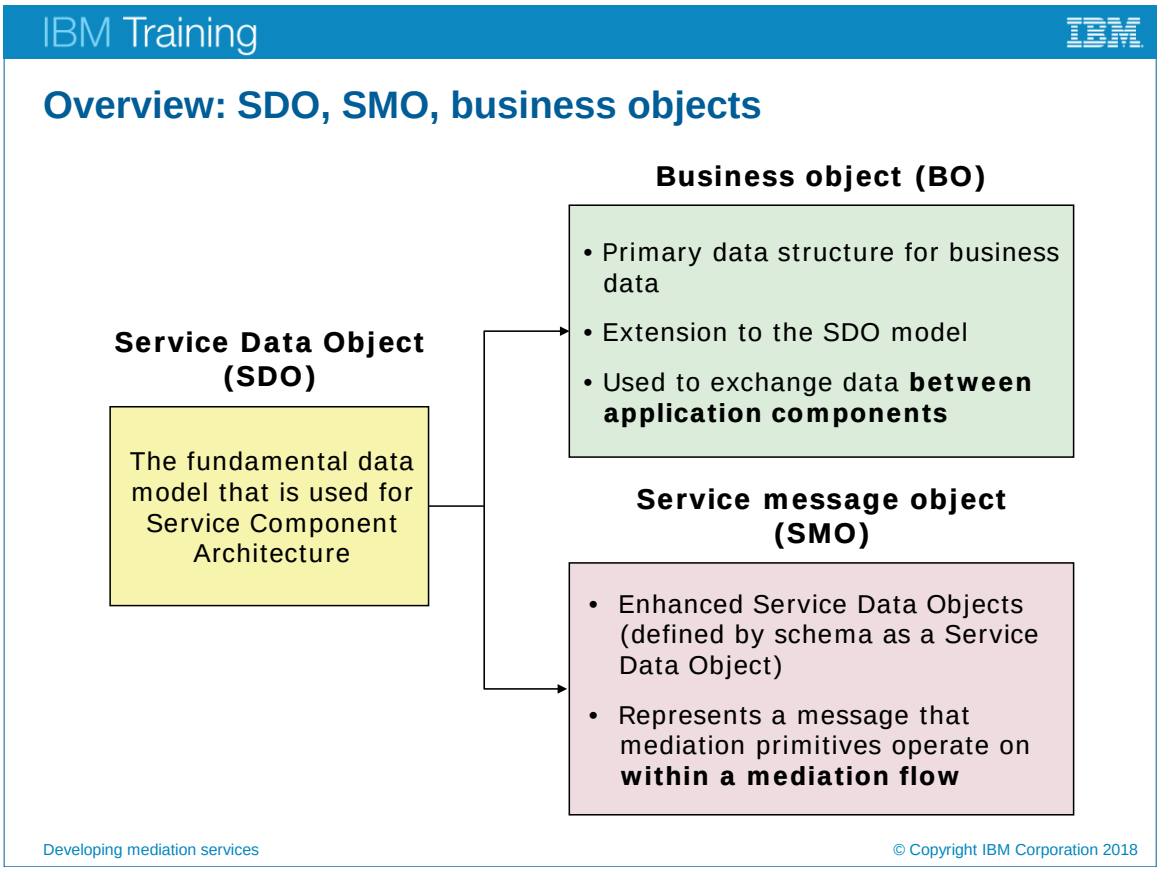
The Integrated Solutions Console is also called the “administrative console” because that is where runtime administrative tasks are done.

Every promoted property has an alias name, which is the name visible to the administrator. Every promoted property belongs to a group. You can create collections of promoted properties for a mediation module. Properties that have the same alias name in the same group are administered together.

In most cases, you want to change the group name and alias name from the default values that are supplied by IBM Integration Designer. This change helps you to identify properties at run time. If you use the default alias names, you are not able to distinguish between the promoted properties of primitives of the same type, if these primitives are assigned to the same group. Promoted properties that have the same alias name and group use the same values.

Promoted properties that are changed through the administrative console affect all messages that are processed within the mediation flow. Dynamic properties allow promoted properties to be overridden on a message-by-message basis. This overriding capability requires the use of mediation policies that are stored in a registry. It also requires the use of a special mediation primitive, the Policy Resolution primitive, which is wired in before the primitive whose properties change.

Service message objects



Overview: SDO, SMO, business objects

Mediations operate primarily on **service message objects**. A service message object consists of three major parts:

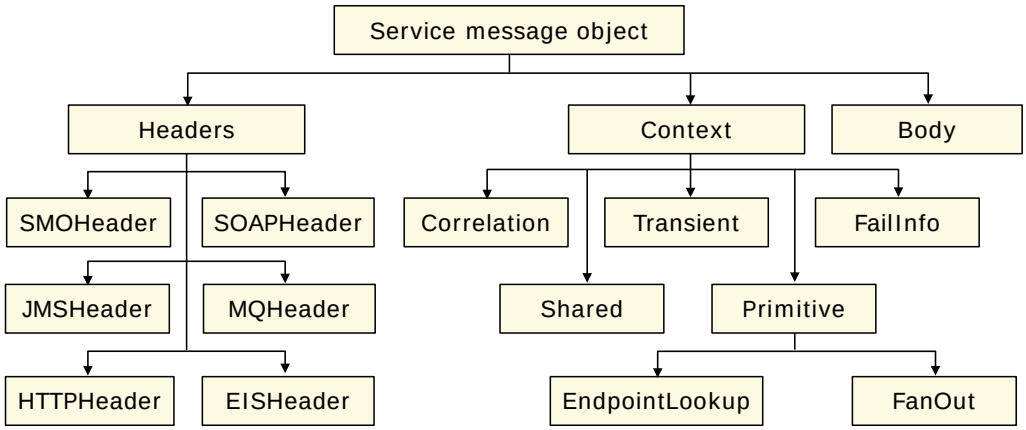
- **Headers** contain information relevant to the transport protocols used to bring the message onto and off the enterprise service bus.
- **Context** contains data about the message as it is propagated from one mediation primitive to the next within a mediation flow.
- **Body** is the data that is sent from the invoking application to the application that is being invoked. Body is encapsulated in a business object. Body is often referred to as the payload of the message. A service message object exists only within a mediation flow.

A **business object** is a structure; it is often a complex data type that contains the actual business data that is being transmitted through or manipulated by the mediation flow. Service Data Object is the standard conceptual data model that is used in Service Component Architecture. The concepts of SDO are implemented physically as business objects and service message objects. The service message object is a physical realization or implementation of Service Data Object principles.

IBM Training

Service message object (SMO)

- At the top level, an SMO is composed of headers, context, and body
  - Header is information about the transport protocol that is used to send a message
  - Context is other data specific to the logic of the flow as data passes between primitives (such as failure information)
  - Body (optional) is the application data (payload) of the message that contains the input or output values of the operation



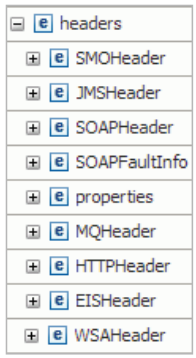
Developing mediation services

Service message object (SMO)

The service message object contains the transport headers, context information that is used during the lifetime of the SMO when it flows through a mediation, and the message data itself, the body (payload). The payload is usually the business object. The body is not required; it is possible for an SMO to contain only headers and context information.

IBM Training

SMO structure: Headers



The headers might include:

- **SMOHeader**: Information about the message (for example, the message ID and the SMO version)
  - SMO header is **always** present
- **JMSHeader**: Contains a JMS header
  - Used when a JMS import or export binding exists
- **SOAPHeader**: Contains SOAP header information
  - Used when a web service import or export binding exists
- **SOAPFaultInfo**: Contains information about SOAP faults
- **properties**[ ]: Arbitrary list of name-value pairs (for example, JMS user properties)
- **MQHeader**: Contains md (MQMD), control (format information of the message body), and other headers (like RFH and RFH2)
- **HTTPHeader**: Contains HTTP headers when an HTTP import or export binding exists
- **EISHeader**: Present when the email, flat file, or FTP adapters are used
  - Different headers exist for each adapter
- **WSAHeader**: Used when an export is configured with JAX-WS binding to provide WS-Addressing support

Developing mediation services

SMO structure: Headers

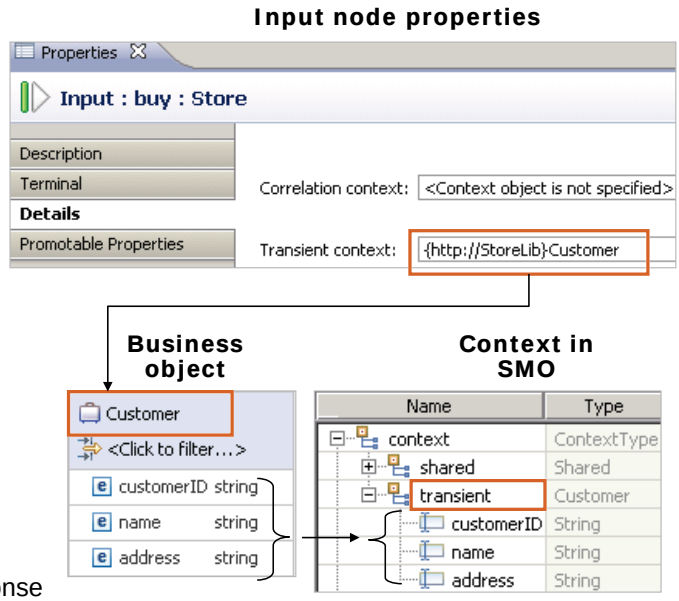
The SMO headers contain information that originates from a specific import or export binding (a binding specifies the message format and transport protocol details). Messages can come from a number of sources, so the SMO can carry different kinds of message headers.

IBM Training

SMO structure: Correlation and transient contexts

The context includes the **correlation** and **transient** contexts

- Both of them are:
  - Used to pass application data between mediation primitives
  - XSD-defined data objects
  - Specified on the input node properties of the mediation flow
- **Correlation** context maintains data across a request/response flow
- **Transient** context:
  - Exists only while the flow is running
  - Maintains data only during one direction (request or response), but the same data object definition is used for both the request and the response



Developing mediation services

SMO structure: Correlation and transient contexts

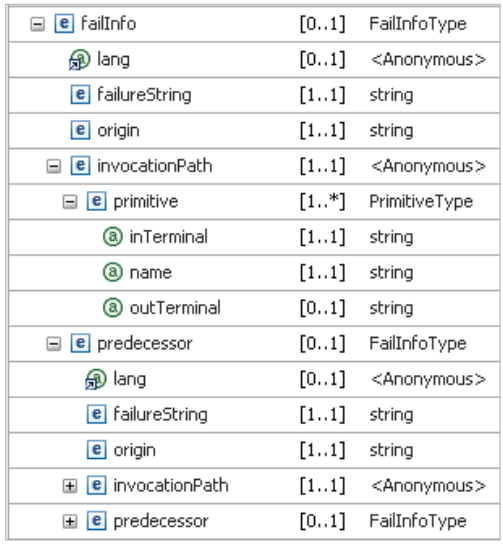
Using the service message object **context**, mediation primitives can pass data that is not part of the message payload, between themselves. Several context types are available.

IBM Training

SMO structure: FailInfo context

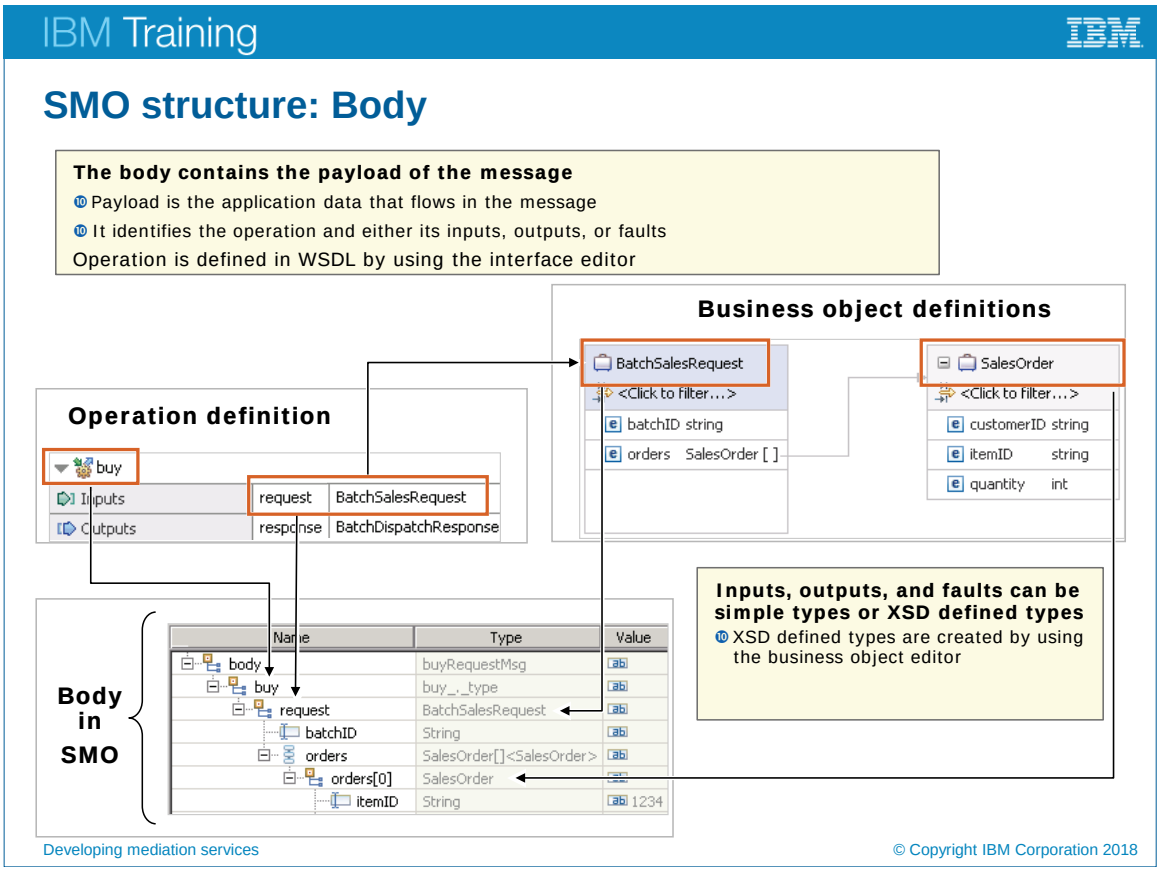
- The **context** also includes the `failInfo` type
  - Contains failure information
  - Added to an SMO when it flows across the fail terminal of a primitive
- The information that is provided includes:

failureString	Describes the failure
origin	Mediation primitive in which failure occurred
invocationPath	The flow that is taken through the mediation
predecessor	Previous failure



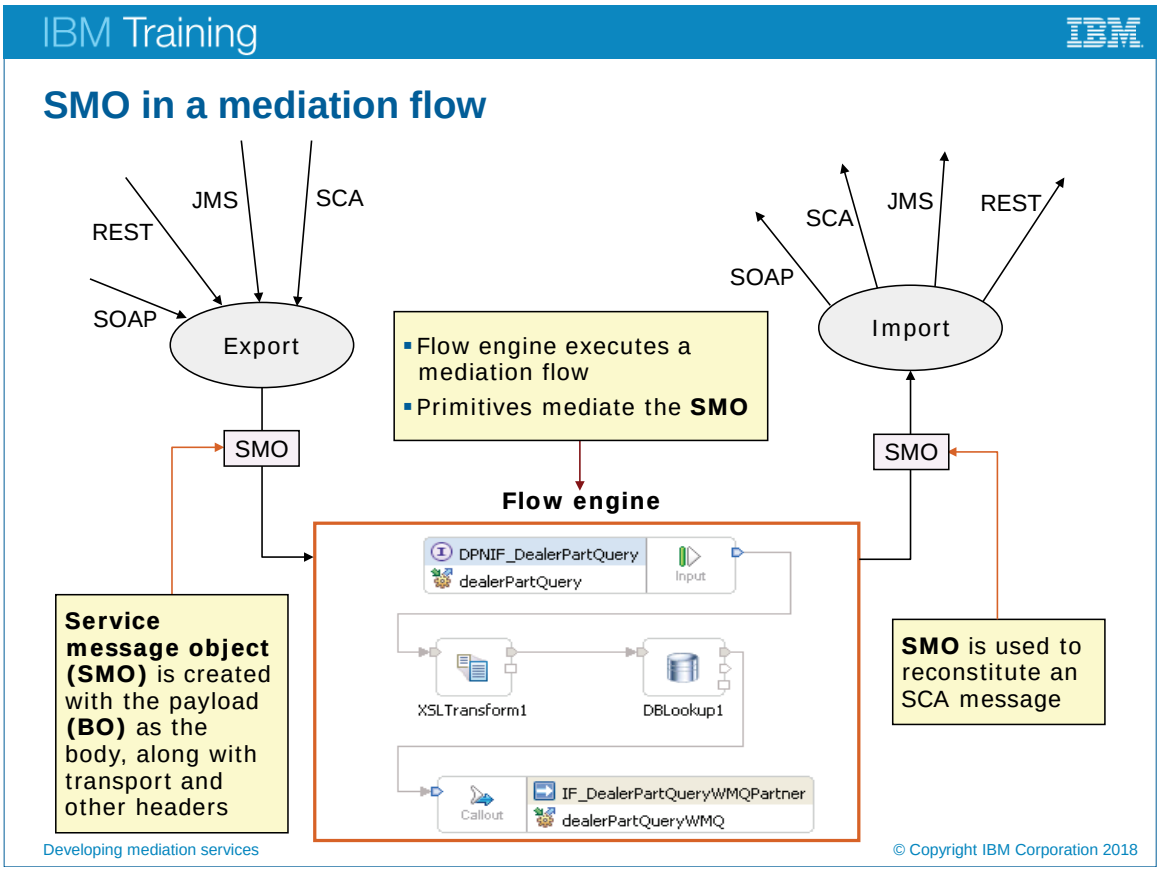
Developing mediation services

SMO structure: FailInfo context



SMO structure: Body

The body of the SMO message contains the actual payload of the message. The body of the message identifies the operation on the data, along with its input message type, output message type, or fault message type, depending on which operation is specified.



SMO in a mediation flow

This visual shows the various forms that data takes through a mediation flow. In this case, the message enters as a JMS message and leaves as a SOAP message.

- The export receives the incoming message, along with the information about the transport protocol by which it is received. The artifacts of the export, including the bindings, interface, and business object, are used to construct the service message object (SMO). Business objects represent the data that flows between each service. Whether the interface that is associated with a particular component is defined as a Java interface or a WSDL port type, the input and output parameters are represented by using business objects.
- When the message is passed into the mediation flow, it is converted to a service message object. The service message object is, again, composed of the transport (and other) headers, plus the actual data (“payload”) that is sent from the invoking application. That data is stored in the service message object as a business object.

- As the message (as an SMO) is propagated through the mediation flow, the various mediation primitives access and modify the elements of the SMO. Some primitives examine the header structure, while others might modify the payload in the business object. Still others update the transport headers in preparation for the message that is being sent to another service that uses a different transport protocol from the one on which the message was received.
- When the message reaches the end of the mediation flow (for example, when the mediation invokes another service), the artifacts of the import are used to reconstitute the SMO into the appropriate native message type. The data in the outgoing message is derived from the payload of the SMO. The transport protocol uses the headers in the SMO to know how to deliver the message to its next destination.

IBM Training

Manipulating SMOs (1 of 2)

- Three ways to access and manipulate SMOs: XPath, XSL, and Java
- XPath expressions
  - Primary mechanism for accessing the SMO
  - Used in some form by all of the mediation primitives
  - Identify elements to read or update the conditional expressions
- XSL stylesheets
  - Used by the Mapping mediation primitive
  - Normally used to modify an SMO type within a flow
  - Can also be used to manipulate the SMO content without changing the type
- Java code
  - Used by the Custom Mediation primitive
  - SMO is accessed through generic Service Data Object APIs (loosely typed), or the SMO APIs (“strongly typed”)
  - Can access and update content
  - Can also modify the SMO type

Manipulating SMOs

The ability to access or modify the contents of the SMO is a fundamental notion in mediation programming. SMOs can be manipulated in several ways in IBM Process Server.

IBM Training

Manipulating SMOs (2 of 2)

- A set of Java services is available to allow business objects to be created and manipulated
- A **shallow** copy uses object references and does not copy primitive data types
  - In a **deep** copy, the objects themselves get copied
- These services are part of the `com.ibm.websphere.bo` package
- Classes include:
  - **BOFactory**: Creates instances of business objects
  - **BOXMLSerializer**: Converts to a business object from a stream, or writes the content of a business object in XML format to a stream
  - **BOCopy**: Methods that make copies of business objects (“deep” and “shallow” semantics)
  - **BODataObject**: Allows access to the data object aspects of a business object, such as the change summary, the business graph, and the event summary
  - **BOXMLDocument**: Allows the manipulation of the business object as an XML document
  - **BOChangeSummary** and **BOEventSummary**: Access and manipulate the change summary and event summary portion of a business object
  - **BOEquality**: Determines whether two business objects contain the same information, both “deep” and “shallow” equality

Developing mediation services

Classes include:

- **BOFactory**: Creates instances of business objects
- **BOXMLSerializer**: Converts to a business object from a stream, or writes the content of a business object in XML format to a stream
- **BOCopy**: Methods that make copies of business objects (“deep” and “shallow” semantics)
- **BODataObject**: Allows access to the data object aspects of a business object, such as the change summary, the business graph, and the event summary
- **BOXMLDocument**: Allows the manipulation of the business object as an XML document
- **BOChangeSummary** and **BOEventSummary**: Access and manipulate the change summary and event summary portion of a business object
- **BOEquality**: You can use BOEquality to determine whether two business objects contain the same information, both “deep” and “shallow” equality
- **BOType** and **BOTypeMetaData**: Materialize instances of the commonj.sdo type; you can manipulate the associated metadata
- **BOInstanceValidator**: Validates the data in a business object to determine whether it conforms to the XSD

IBM Training

Unit summary

- Describe the role of mediation services in IBM Process Server
- Define the concept of mediation modules
- Describe how to create mediation flows in IBM Integration Designer
- Describe the role of SMOs in mediations
- Explain the structure of SMOs

Unit summary

IBM Training

Checkpoint questions

1. True or False: A mediation primitive contains the logic to transform the content of a message.
2. True or False: The operation connections in the Overview perspective of the mediation flow editor display the mappings between the interfaces and operations of a mediation.
3. True or False: Mediation primitive terminals are associated with the message type that the primitive is processing, in most cases.
4. True or False: Input and output messages to the mediation primitives are represented as service message objects.
5. True or False: XPath expressions can be used to manipulate an SMO.

Checkpoint questions

IBM Training

Checkpoint answers

1. True.
2. False: They show the mapping between interfaces and references.
3. True.
4. True.
5. True.

Checkpoint answers

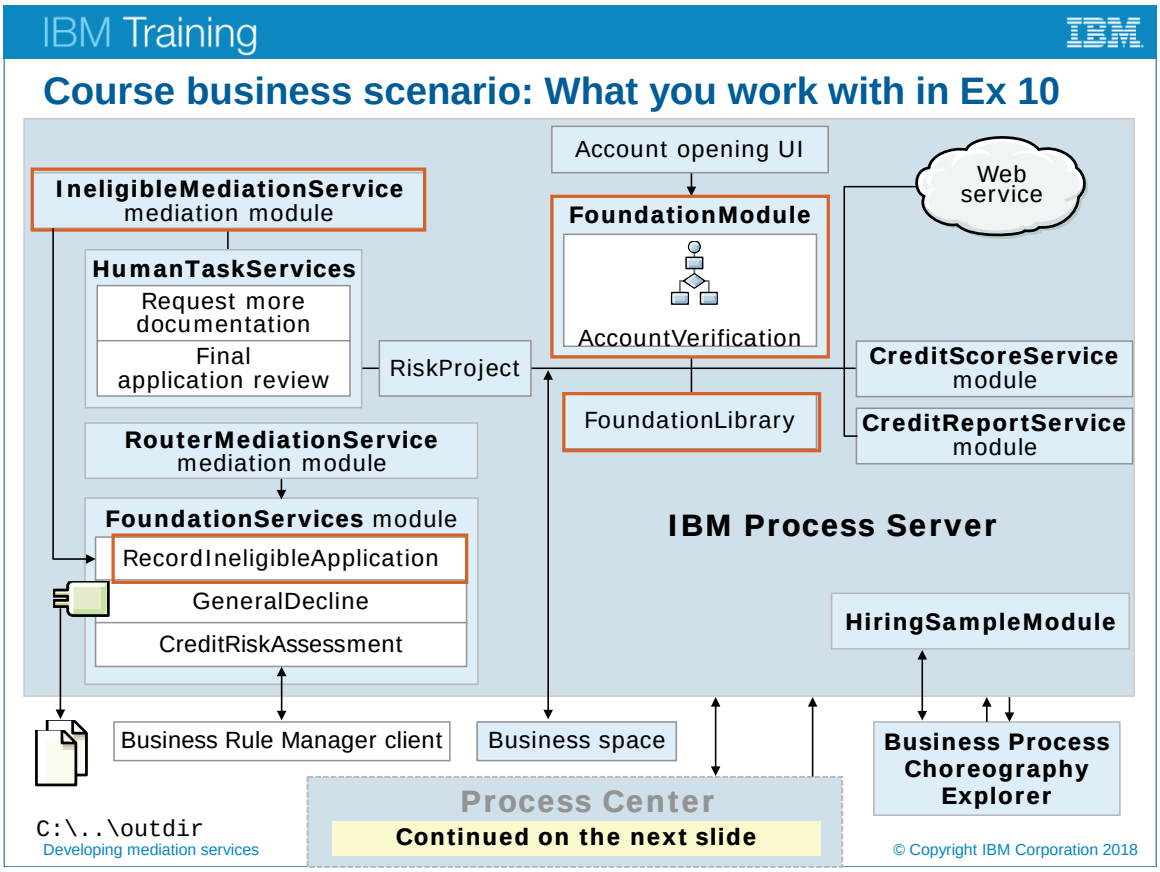
IBM Training

Exercise 10: Creating mediation services, part I

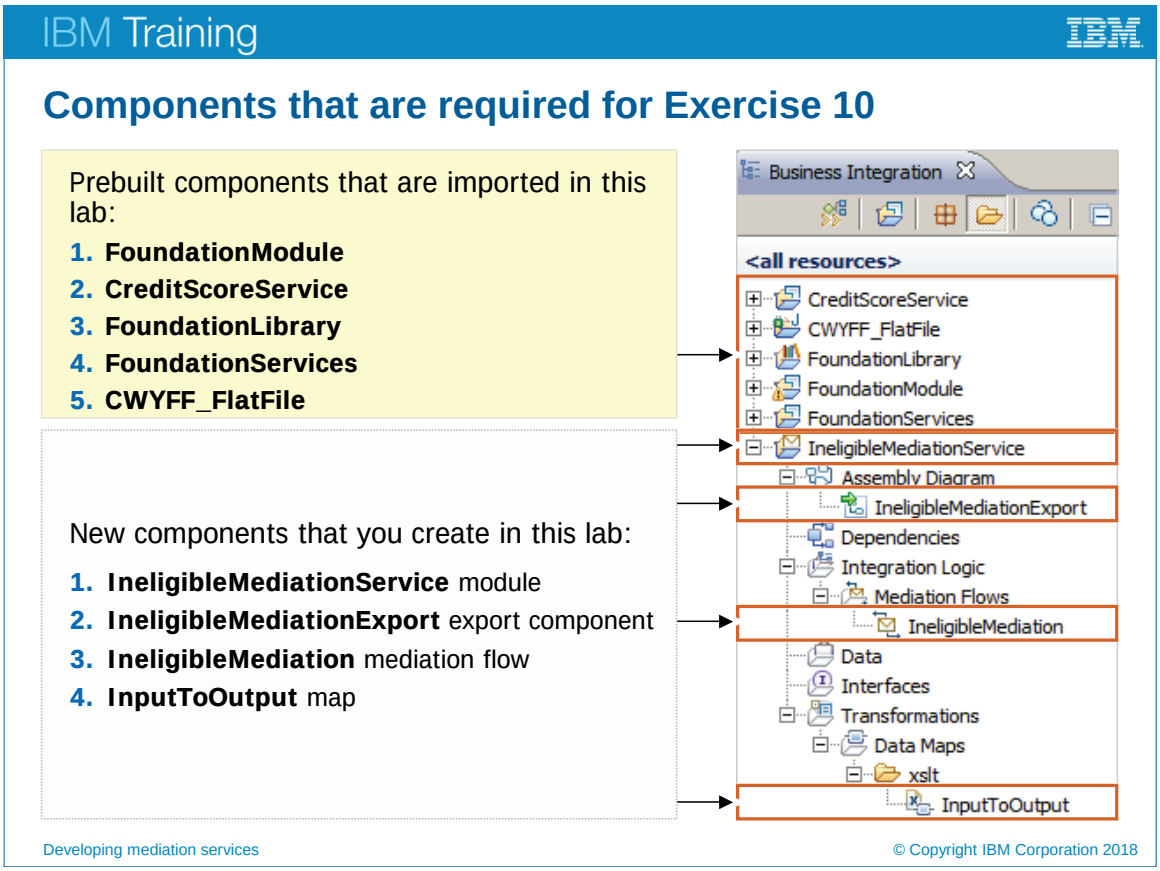
After completing this exercise, you should be able to:

- Create a mediation module that contains a Mapping primitive
- Define an XML data map
- Test a mediation module that contains a Mapping primitive

Exercise 10: Creating mediation services, part I



Course business scenario: What you work with in Ex 10

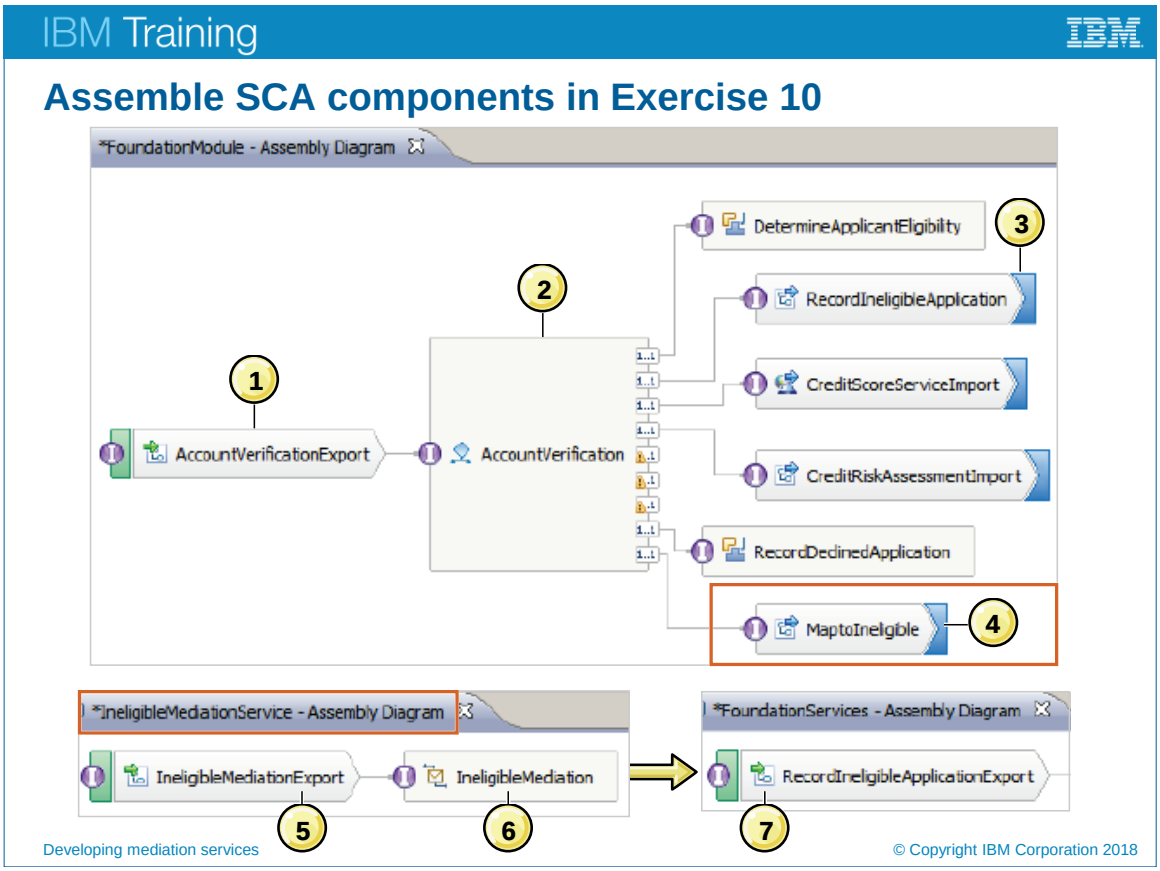


Components that are required for Exercise 10

In this exercise, you implement a mediation module that is named IneligibleMediationService, which transforms messages between the AccountVerification process and the RecordIneligibleApplication service (which archives ineligible applications). Suppose the data input for the RecordIneligibleApplication service is different from the CustomerApplication input that the AccountVerification process uses. Under this scenario, the RecordIneligibleApplication component does not read the data and no output message is sent. To fix this problem, you must transform the messages that are passed between the services. This problem is a common challenge that your application must be prepared to resolve, and IBM Integration Designer solves it by creating a transformation between the two messages.

You implement a Mapping primitive inside a mediation flow to transform the messages by using a data map. After implementing the transformation in the mediation flow, you assemble the application by wiring the mediation service to the AccountVerification process.

Finally, after creating the flat file adapter component, you assemble the SCA components in IBM Integration Designer and then test the application.



Assemble SCA components in Exercise 10

The following steps are illustrated in the diagram:

1. The AccountVerificationExport component exposes the AccountVerification business process.
2. When the application is ineligible, the AccountVerification process needs to record the ineligible application in the database and terminate the process. It calls the RecordIneligibleApplication import component.
3. The RecordIneligibleApplication import component is used to call the application or service outside FoundationModule. In this scenario, it calls the RecordIneligibleExport component in the FoundationServices module.
4. The MaptoIneligible import invokes the IneligibleMediationExport component in the IneligibleMediationService mediation module.
5. The IneligibleMediationExport exposes the mediation flow by calling the IneligibleMediation component.
6. The IneligibleMediation component transforms a CustomerApplication into an IneligibleApplication before invoking the RecordIneligibleApplicationExport component to archive the application.

7. The RecordIneligibleExport component calls the RecordIneligibleApplication Java component. The RecordIneligibleApplication component is a Java component that creates the output message.

“Account verification recorded this application as ineligible for the customer <company name>” is going to be recorded to the system. In this scenario, it calls the FlatFileOutboundImport component. The FlatFileOutboundImport component writes the output message to the file system. In this scenario, it writes to a text file in the `C:\IneligibleAppArchive\outdir` output directory.

Exercise 10: Creating mediation services, part I

Purpose:
In this exercise, you create a mediation module and build a simple XML data map.

Mediation flows intercept and modify messages that are passed between services (providers) and clients (requesters) that want to use those services. Implement mediation flows between services to process the messages that are being passed between them. Mediation flows provide the logic that processes the messages.

For example, mediation flows can be used to find services with specific characteristics that a requester is seeking. It can also be used to resolve interface differences between requesters and providers. For complex interactions, mediation primitives can be linked sequentially. Typical mediations include:

- Transforming a message from one format to another so that the receiving service can accept the message
- Conditionally routing a message to one or more target services; the routing is based on the contents of the message
- Augmenting a message by adding data from a data source

Mediation service applications are assembled and deployed as one or more mediation modules. Mediation modules are deployable units that contain mediation flows.

A mediation module can have the following parts:

- Mediation flow components
- Imports that identify service providers and their interfaces
- Exports that expose the mediation module to service requesters
- Java components

Requirements

Completing the exercises for this course requires a lab environment. This environment includes the exercise support files, IBM Process Designer, IBM Process Center, and IBM Integration Designer test environment.

This exercise stores data in a local directory. They were created in a previous exercise. You must make sure that the directories `C:\IneligibleAppArchive\outdir` and `C:\IneligibleAppArchive\staging` are present. If they are not, you must create them.

Exercise Instructions

In this exercise, you implement a simple mediation module that is named `IneligibleMediationService`, which transforms messages between the AccountVerification process and the RecordIneligibleApplication service (which archives ineligible applications). The data input for the RecordIneligibleApplication service differs from the CustomerApplication input that the AccountVerification process uses. As such, you must transform the messages that are passed between the services.

You implement a Mapping primitive inside a mediation flow to transform the messages by using a data map. After implementing the transformation in the mediation flow, you assemble the application by wiring the mediation service to the AccountVerification process.

Part 1. Create a mediation module that contains a Mapping primitive.

In this portion of the exercise, you create a mediation module that is named `IneligibleMediationService`. This module transforms a CustomerApplication into an IneligibleApplication before invoking the RecordIneligibleApplicationExport component to archive the application. Using a Mapping primitive, you implement an XML data map to transform the data in the body of the service message objects that are passed between the services.

To create the IneligibleMediationService:

1. Open the Exercise 10 workspace.

1. On your desktop, open the **Exercise Shortcuts** folder.
2. Double-click the **Exercise 10** shortcut. Allow Integration Designer a few moments to build the workspace. You can view the workspace build status at the lower right corner of Integration Designer. Wait until the status reaches 100%, at which point the workspace is built, and the status progress bar disappears.
3. If you get a message that the server is already set to publish, then click **OK**.
4. Close the **Getting Started** tab.

2. Create a mediation module that is named `IneligibleMediationService`.

The target runtime environment for the module is **IBM Process Server v8.6**, and the module has a dependency on FoundationLibrary. The mediation component in the module is named `IneligibleMediation`.

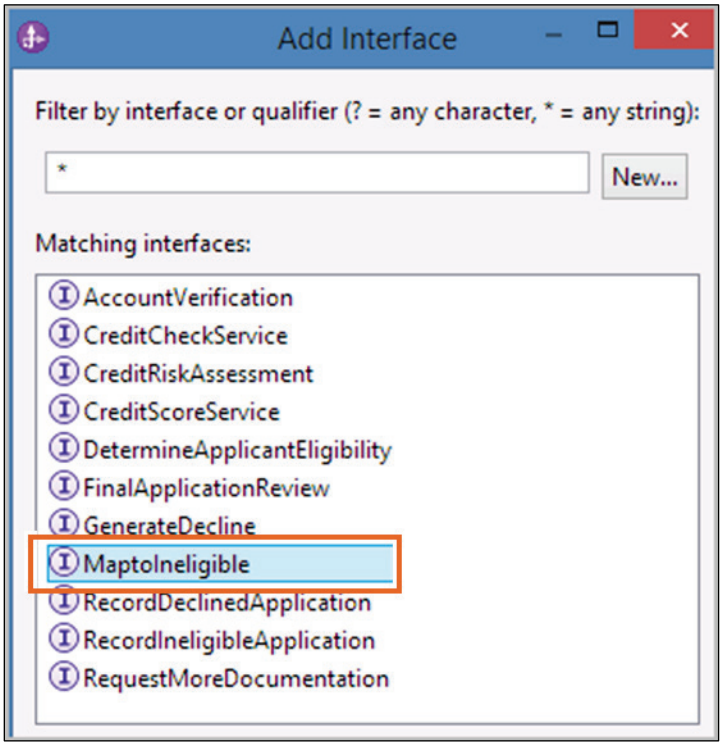
1. Click File > New > Mediation Module from the menu options.

2. At the **Create a Mediation Module** pane, enter the following information.
   - Type `IneligibleMediationService` in the **Module name** field. This action automatically places `IneligibleMediationService` in the mediation component Name field.
   - Change the **Name** of the mediation component from `IneligibleMediationService` to: `IneligibleMediation`
   - Select **IBM Process Server v8.6** from the **Target runtime environment**.
3. Accept the remaining default options and click **Next**.
4. Select **FoundationLibrary** check box to include it as a dependency.

5. Click **Finish**.

2. Add the MaptoIneligible interface to the IneligibleMediation mediation flow component.

1. Right-click the **IneligibleMediation** component and click **Add > Interface** from the menu.
2. In the Add Interface dialog box, select **MaptoIneligible**.

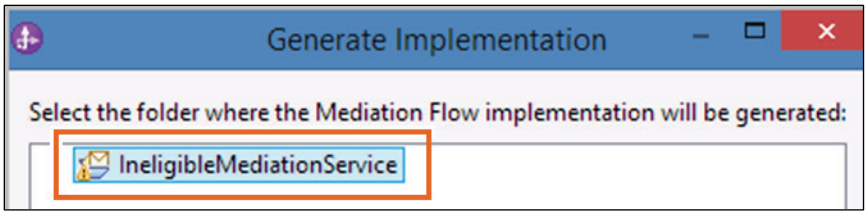


3. Click **OK**.

   Use the **Blank Mediation Flow** template to generate the implementation for the **IneligibleMediation** mediation flow component.

4. Right-click the **IneligibleMediation** component and click **Generate Implementation** from the menu.

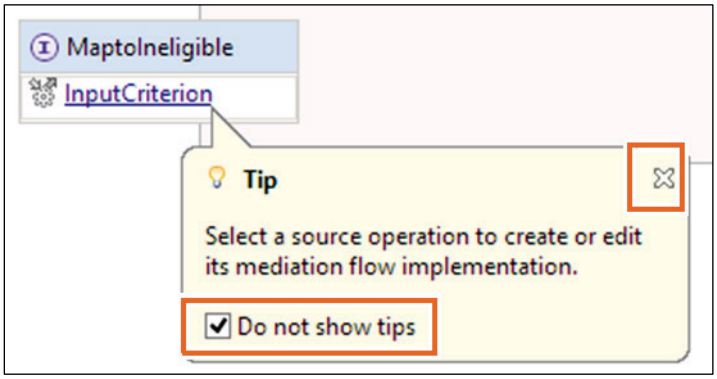
5. In the Generate Implementation dialog box, select the **IneligibleMediationService** folder.



6. Click **OK**.

   The mediation flow editor opens.

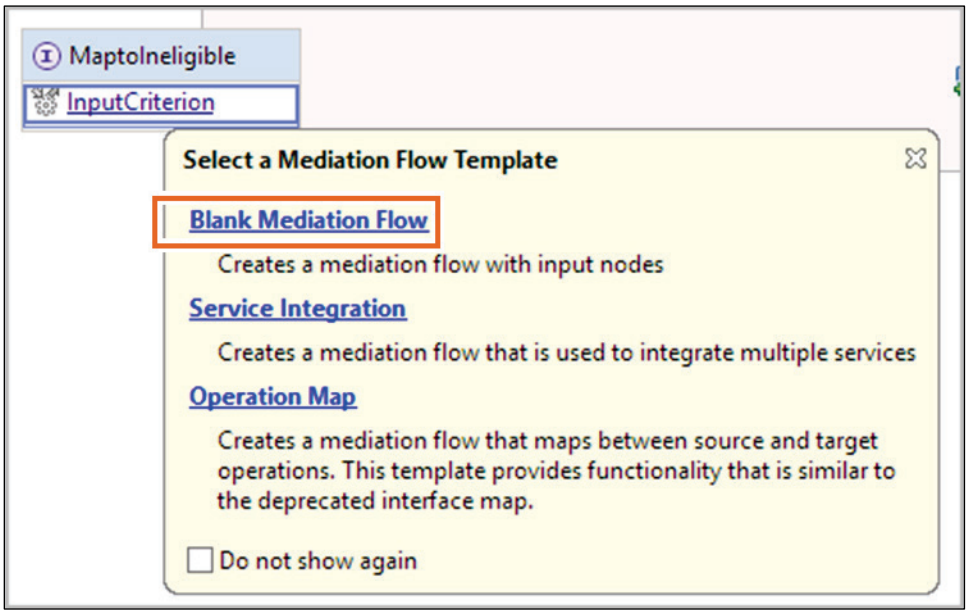
7. If a Tip dialog box opens in the mediation flow editor, select **Do not show tips** in the dialog box and close it.



Part 2. Define an XML data map.

1. Create an XML data map to transform the input parameters into the output.

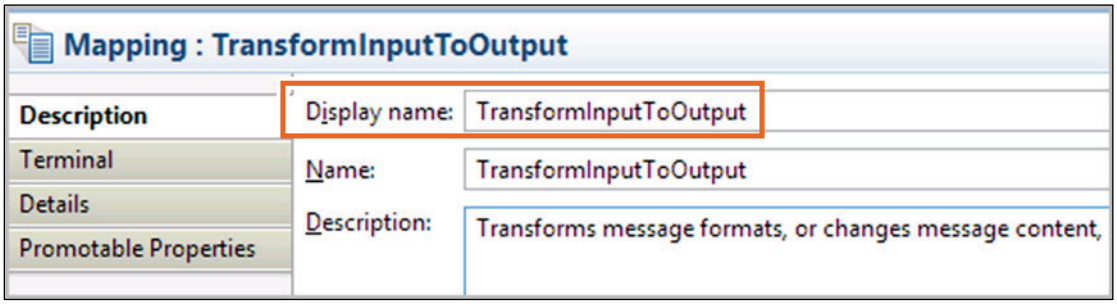
1. In the Operation connections section of the editor, click the **InputCriterion** operation in the **MaptoIneligible** interface.
2. In the "Select a Mediation Flow Template" dialog box, click the **Blank Mediation Flow** link.



3. The generated mediation request flow consists of an **Input** node and an **Input Response** node. You might find it necessary to scroll to the right to see the **Input Response** node.

2. You will add a Mapping primitive to the request flow that is named: TransformInputToOutput

1. In the palette, expand **Transformation** and select **Mapping**.
2. Click the blank space on the canvas between the Input node and the Input Response node to add the primitive to the diagram.
3. Switch to the **Description** tab in the **Properties** view.
4. Change the **Display Name** to: `TransformInputToOutput`



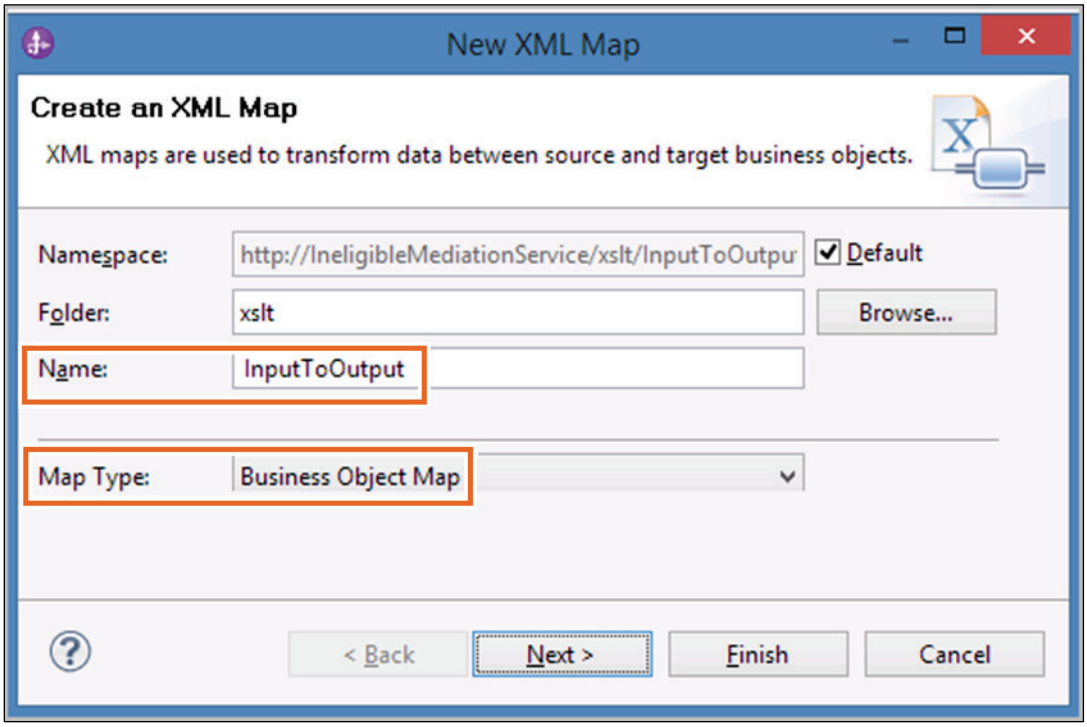
5. Save your changes.

   The request flow resembles the following figure:

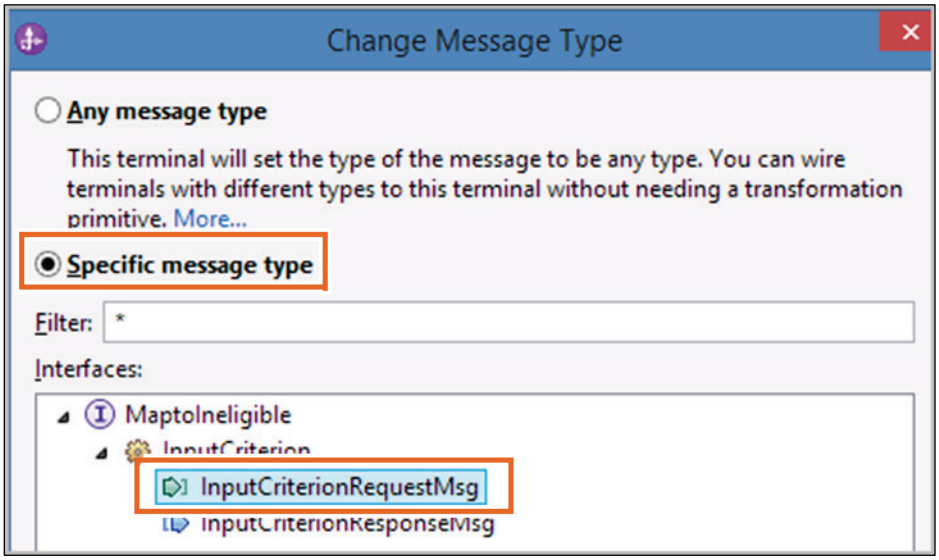
3. Create the XML data map that the Mapping primitive uses to transform the body of the CustomerApplication input to the body of an IneligibleApplication output.

The input message for the map is `InputCriterionRequestMsg`. The output message for the map is `InputCriterionResponseMsg`.

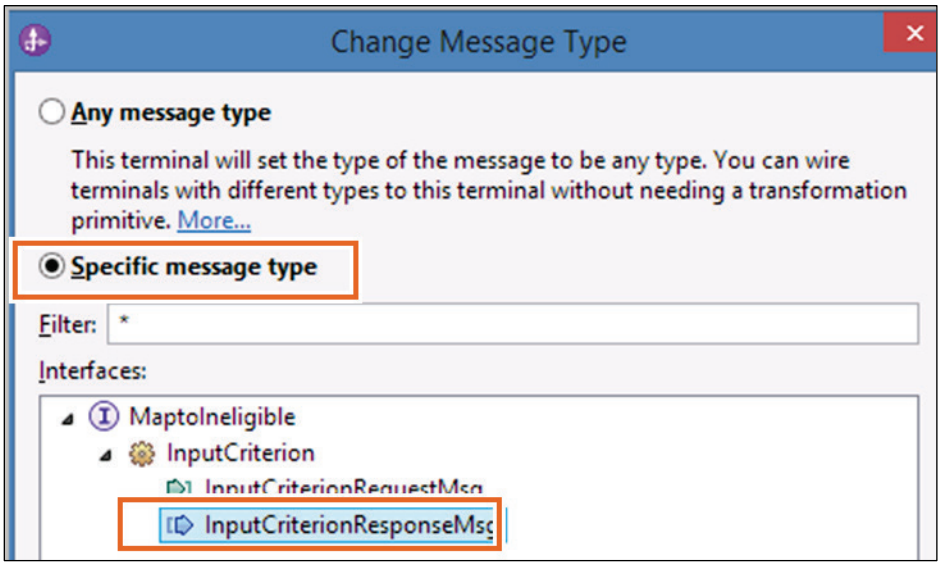
1. With **TransformInputToOutput** selected, switch to the **Details** tab in the **Properties** view.
2. For the **Mapping File** field, click **New**.
3. In the “Create an XML Map” pane, enter the following information:
   - Verify that `xslt` is the value in the **Folder** field.
   - Type `InputToOutput` in the **Name** field.
   - Select **Business Object Map** in the **Map Type** field.



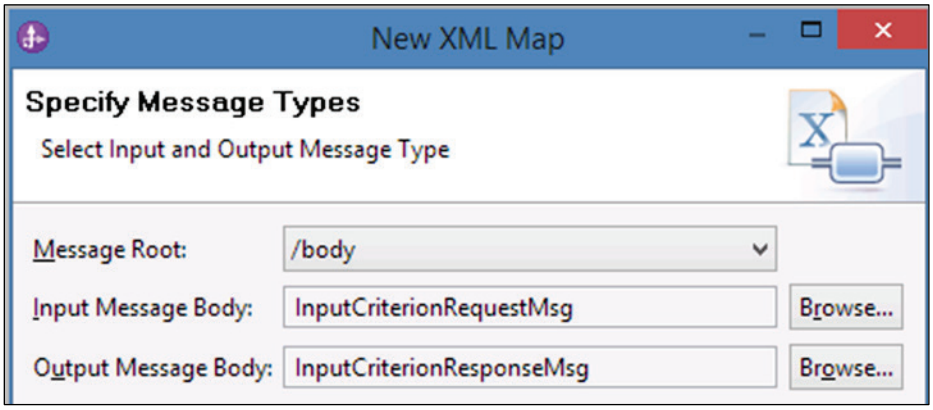
4. Before you click **Next**, view the different options available in the **Map Type** field. They are `XSLT 1.0 map`, `XSLT 2.0 map`, and `Business Object Map`. You can use the Mapping mediation primitive to switch between XSL stylesheets and business object maps. When performance is important, you can set the map generation type to generate a business object map instead of an XSLT. Alternatively, you can also generate a business object map instead of an XSLT when the advanced functions that XSLT offers are not required.
5. Click **Next**.
6. In the Specify Message Types dialog box, take the following actions:
   - For **Message Root**, select **/body**.
   - Click **Browse** by **Input Message Body**, select **Specific message type**, and select **InputCriterionRequestMsg**.



- Click **OK**.
- Click **Browse** by **Output Message Body**, select **Specific message type**, and select **InputCriterionResponseMsg.**



- Click **OK**.
- The Specify Message Types pane resembles the following figure:



- Ignore any warnings at the bottom of the dialog box and click **Finish** to open the **InputToOutput** map in the mapping editor.

Note: The Message Root field identifies the part of the message that you want to map in the XML mapping editor.

- / (root) transforms the entire message and is used to map SOAP attachments
- `/body` transforms the message body
- `/headers` transforms the message headers
- `/context` transforms the message context

4. Implement the map definition.

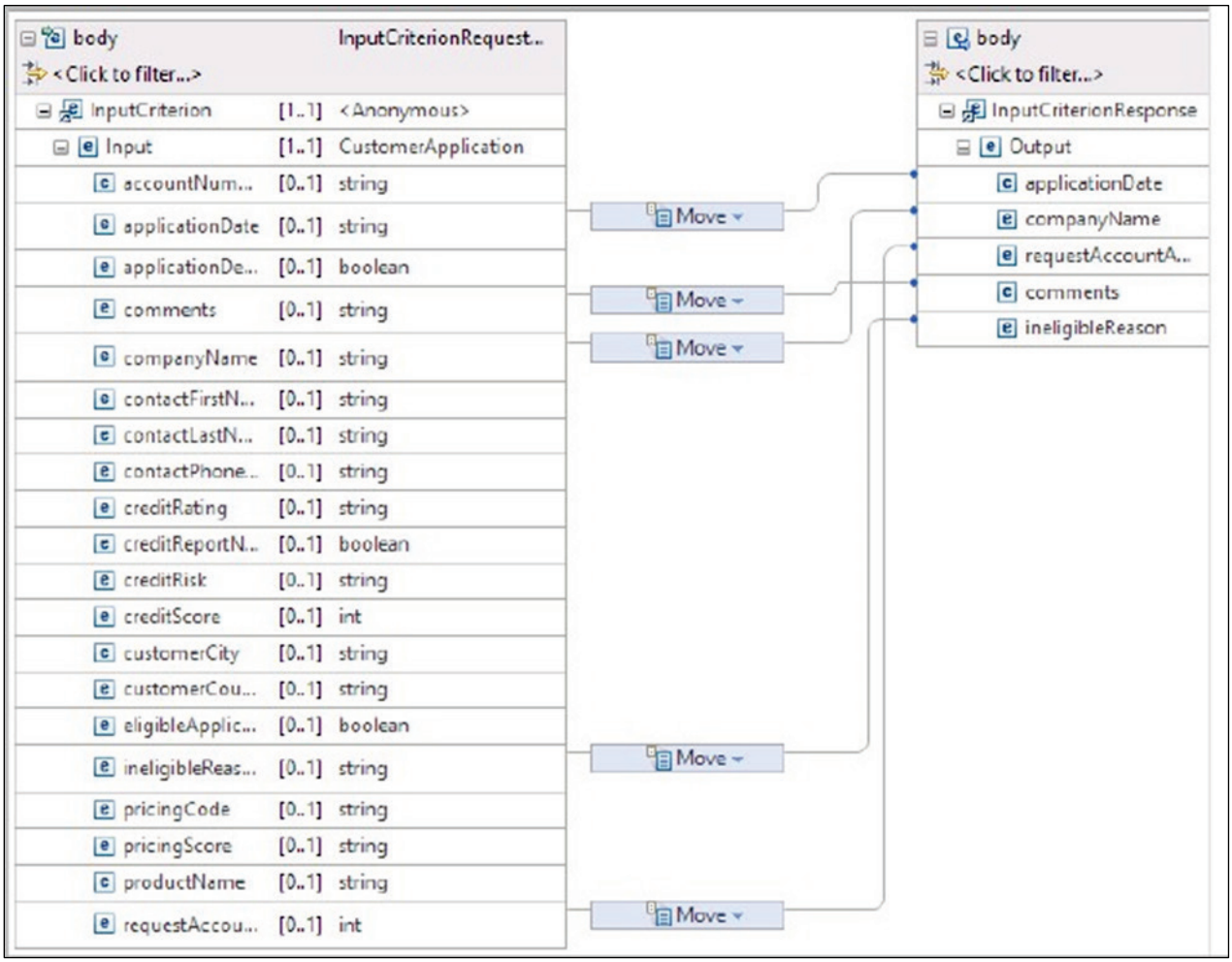
The `applicationDate`, `companyName`, `requestAccountAmount`, `comments`, and `ineligibleReason` elements in the input service message object (SMO) are mapped to the corresponding target elements in the output SMO. All transformations are **Move** transformations.

1. In the mapping editor, expand **InputCriterion > Input** in the input SMO by clicking the plus (**+**) symbols.
2. In the mapping editor, expand **InputCriterionResponse > Output** in the output SMO by clicking the plus (**+**) symbols.
3. In the body section of the input SMO, right-click **applicationDate** and click **Create Connection** from the menu.
4. Click **applicationDate** in the body section of the output SMO to create the **Move t**ransformation.
5. In the body section of the input SMO, right-click **companyName** and click **Create Connection** from the menu.
6. Click **companyName** in the body section of the output SMO to create the **Move** transformation.

7. In the body section of the input SMO, right-click **requestAccountAmount** and click **Create Connection** from the menu.
8. Click **requestAccountAmount** in the body section of the output SMO to create the **Move** transformation.
9. In the body section of the input SMO, right-click **comments** and click **Create Connection** from the menu.
10. Click **comments** in the body section of the output SMO to create the **Move** transformation.
11. In the body section of the input SMO, right-click **ineligibleReason** and click **Create Connection** from the menu.

12. Click **ineligibleReason** in the body section of the output SMO to create the **Move** transformation.

    The completed XML map resembles the following figure:



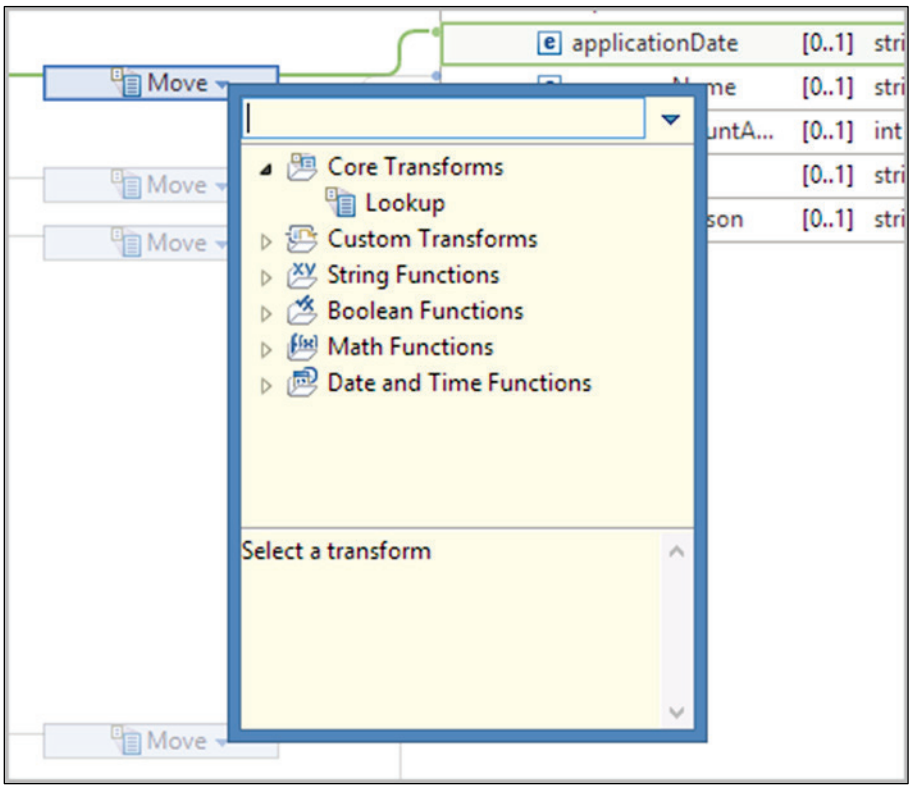
13. Click **File > Save All** from the menu options to save your changes to the map, the mediation flow, and the assembly diagram.

5. You will explore the supported transforms.

1. Verify that you saved your changes before you do the next few steps, or you might get incorrect results.

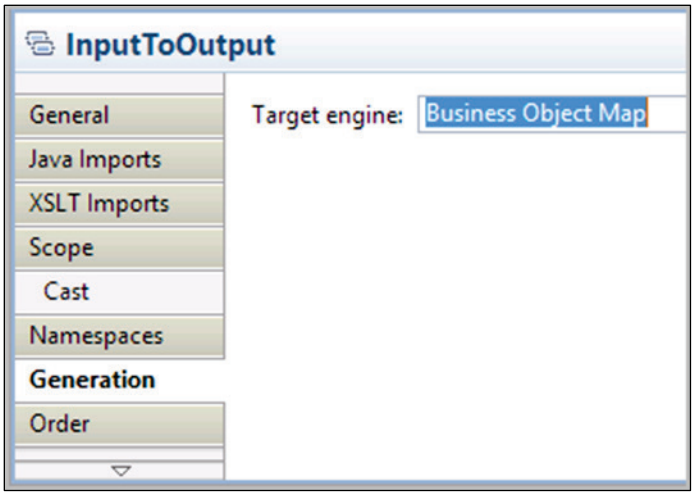
2. Click the down arrow in a Move transform to view the available transforms for the business object map.

   Recall that you set the **map type** to be generated as a **bo map** when you created the Mapping primitive.

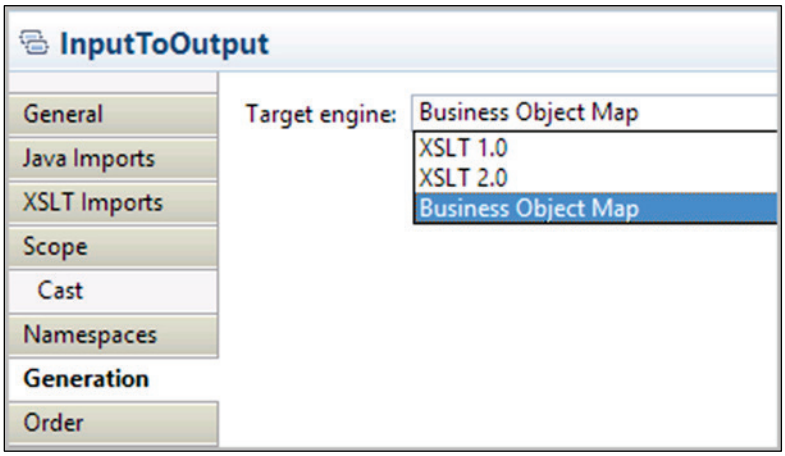


3. Click anywhere in the empty space in the XML map editor and verify that **InputToOutput** is listed in the **Properties** view.
4. Click the **Generation** tab.

   Business Object Map is listed as the Target engine.

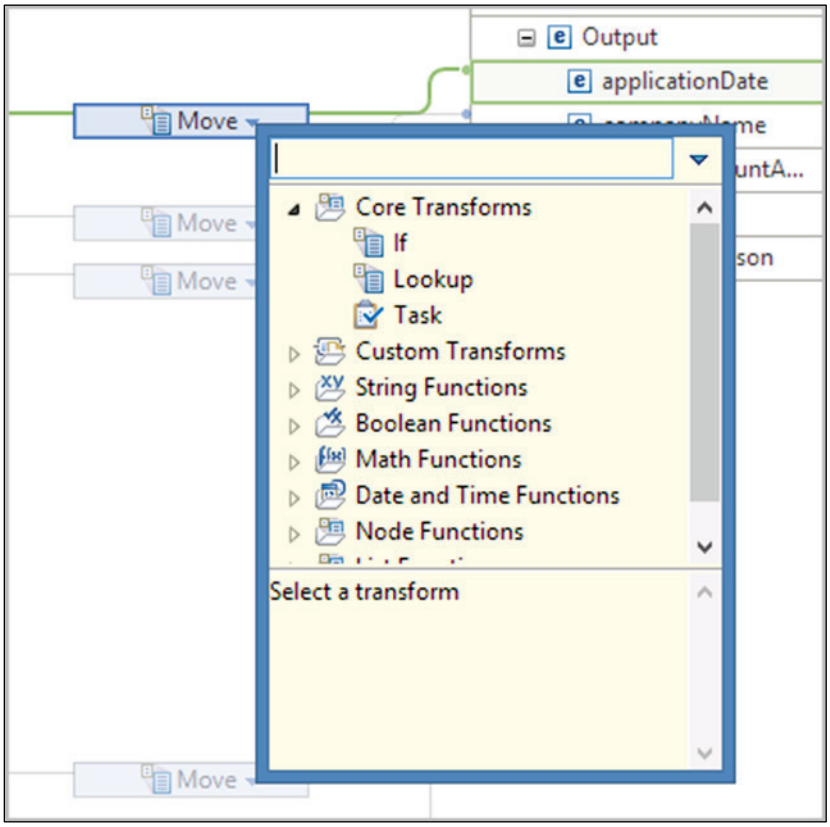


5. Click the **Target engine** list to view the choices available. It offers an option to switch to **XSLT 1.0** or **XSLT 2.0** as the map type.



6. Select **XSLT 1.0** as the **Target engine** type.
7. Click the down arrow in a **Move** transform to view the available transforms.

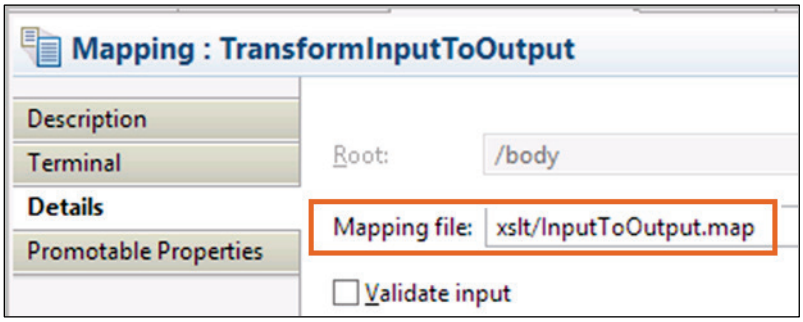
   This time, the list of the transforms changes to support the XSLT map generation. Also, notice that the transforms that a business object map supports are a subset of transforms that XSLT supports. The tool allows an easy switch between the business object map and the XSLT target engine when mapping requirements change. Validations are done for unsupported transforms during the switch.



8. Close the map editor and do not save your changes.

6. Verify the definition of the Mapping primitive.

1. With **TransformInputToOutput** selected in the request flow editor, switch to the **Details** tab in the **Properties** view.
2. Verify that the **Mapping file** is set to: `xslt/InputToOutput.map`

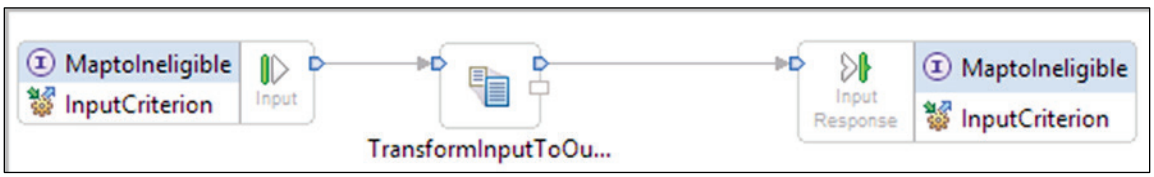


You will assemble the request flow by wiring the Input and Input Response nodes to the Mapping primitive. Wire the **out** terminal of the **Input** node to the **in** terminal of the **TransformInputToOutput** primitive. Wire the **out** terminal of the **TransformInputToOutput** primitive to the **in** terminal of the **Input Response** node.

3. Right-click the **out** terminal of the **InputCriterion** input node and click **Add Connection** from the menu.

   If you find it difficult to right-click the **out** terminal, you can hover over the terminal and grab the orange, circular handle.
4. Click the **in** terminal of the **TransformInputToOutput** primitive to add the connection.
5. Right-click the **out** terminal of the **TransformInputToOutput** primitive and click **Add Connection** from the menu.
6. Click the **in** terminal of the **InputCriterion** input response node to add the connection.
7. Save the **IneligibleMediation** flow.

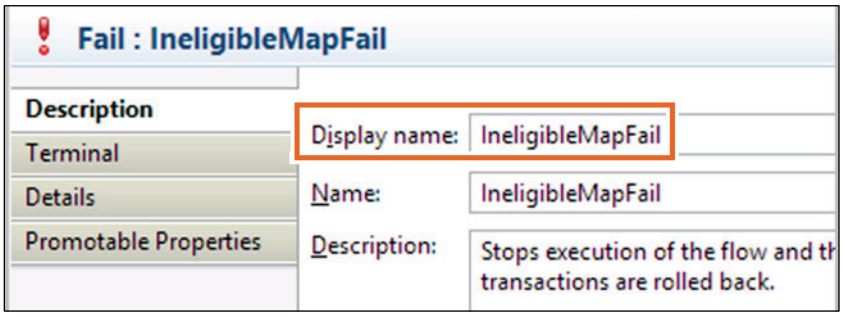
   Your diagram resembles the following figure:



7. Add a Fail primitive to the request flow named: `IneligibleMapFail`

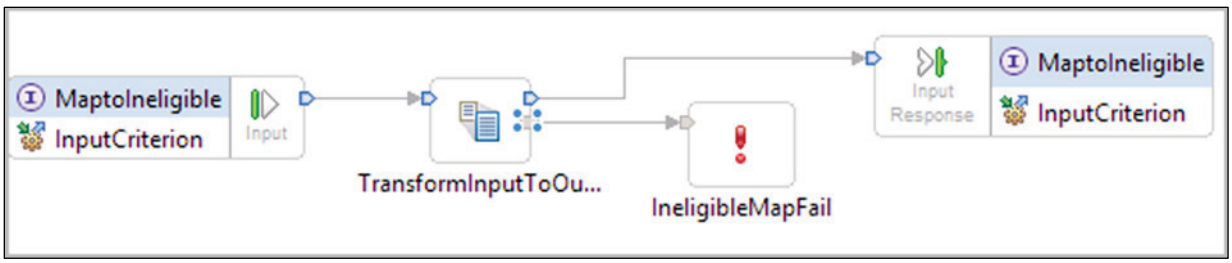
Wire the **fail** terminal of the **TransformInputToOutput** primitive to the **in** terminal of the **Fail** primitive. The fail primitive stops flow execution and throws an exception when a failure is encountered in the map.

1. In the palette, expand **Error Handling** and select **Fail**.
2. Click an empty portion of the diagram to the right of the **TransformInputToOutput** primitive to add the fail primitive to the flow.
3. With the **Fail** primitive selected, switch to the **Description** tab in the **Properties** view.
4. Change the **Display name** to: `IneligibleMapFail`



5. Save your changes.
6. Right-click the **fail** terminal on the **TransformInputToOutput** primitive and click **Add Connection** from the menu.
7. Click the **in** terminal on the **IneligibleMapFail** primitive to add the connection.
8. Save your changes.

   The request flow diagram resembles the following figure:



9. Close the **IneligibleMediation** tab to return to the assembly diagram.

   You do not implement a response flow for this mediation because the flat file adapter processes communications in only one direction: outbound.

8. Create an export with SCA binding for the IneligibleMediation component.

1. In the assembly diagram, right-click **IneligibleMediation** and click **Generate Export > SCA Binding** from the menu.
2. Accept the default export name: `IneligibleMediationExport`

3. Save your changes.
   Your assembly diagram resembles the following figure:

4. Close the assembly diagram and make sure that you have no errors in the Problems view.
   You can ignore any warnings.

Part 3. Test a mediation module that contains a Mapping primitive.

In this portion of the exercise, you wire the IneligibleMediationService to the AccountVerification business process on the FoundationModule assembly diagram. The RecordIneligibleApplication service can be called to archive an ineligible application by using the flat file adapter. However, before that call is made, the CustomerApplication data object used by the AccountVerification business process must be transformed.

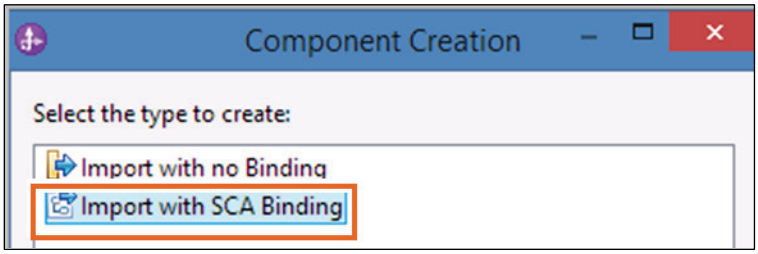
The IneligibleMediationService transforms a CustomerApplication data object into an IneligibleApplication data object that is used by the RecordIneligibleApplication service. The IneligibleMediationService mediation module uses a Mapping primitive (and an XML data map) to transform the data.

To wire the IneligibleMediationService to the AccountVerification business process:

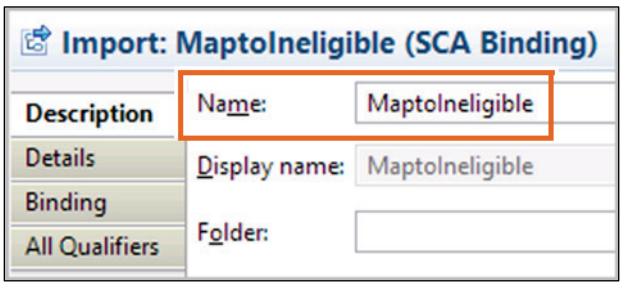
1. Create an import component that is named `MaptoIneligible` on the FoundationModule assembly diagram. The MaptoIneligible import invokes the IneligibleMediationExport component in the IneligibleMediationService mediation module.

1. In the Business Integration view, expand **FoundationModule** and double-click **Assembly Diagram.**
2. In the Business Integration view, expand **IneligibleMediationService > Assembly Diagram**.
3. Drag **IneligibleMediationExport** onto the **FoundationModule** assembly diagram.

4. Select **Import with SCA Binding** and click **OK**.



5. Select the import and switch to the **Description** tab in the **Properties** view.
6. Change the Name to: `MaptoIneligible`



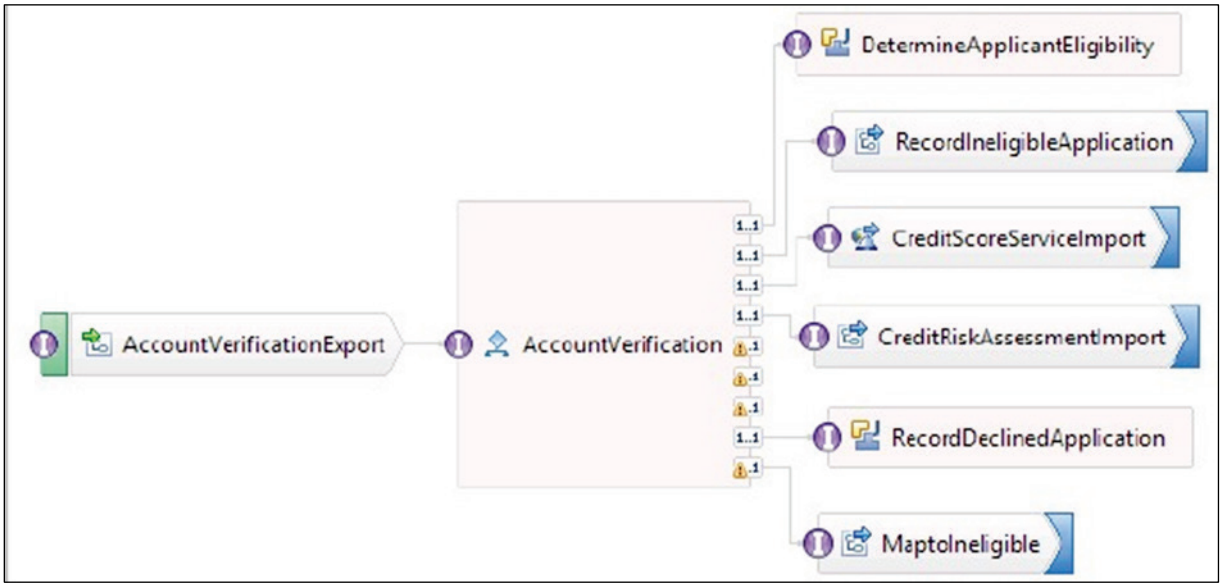
7. Save your changes.

2. Wire the MaptoIneligiblePartner reference on the AccountVerification process component to the newly created MaptoIneligible import.

1. Right-click the **MaptoIneligible** import component and click **Wire to Existing** from the menu.
2. Save your changes.
3. Verify the wiring by hovering over the reference that is wired to the **MaptoIneligible** import.

   The dialog box displays the **MaptoIneligiblePartner** reference.

   Your **FoundationModule** assembly diagram resembles the following visual:



Testing the applications

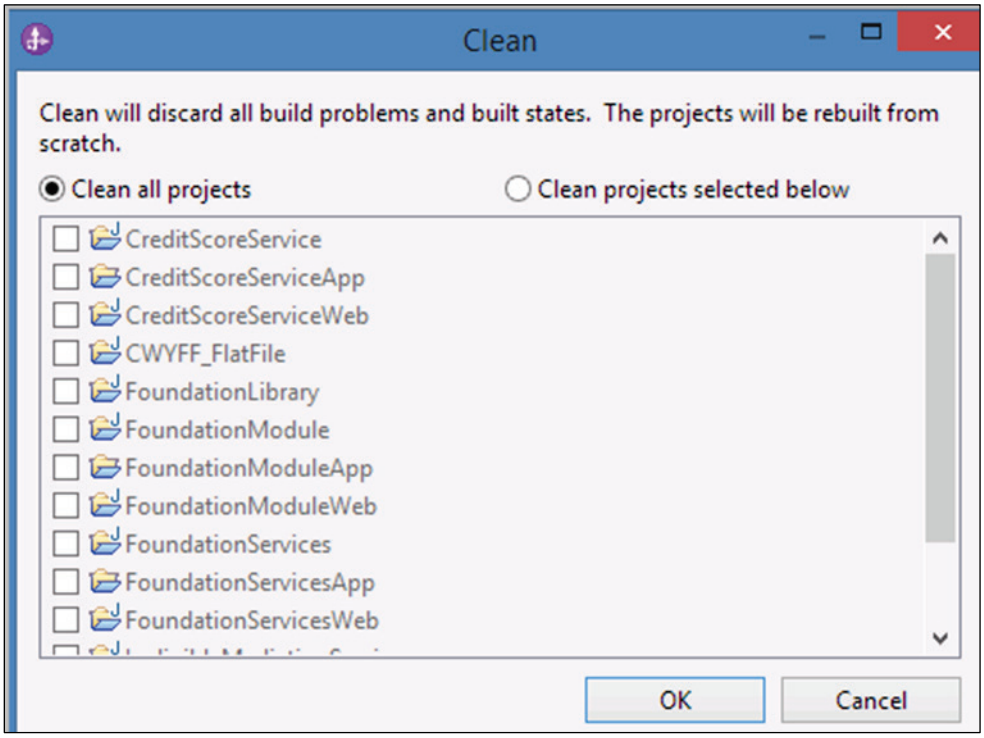
Now that you completed the IneligibleMediationService mediation module implementation, the “ineligible” path through the AccountVerification business process is complete. If an application is determined to be ineligible for an account, the eligibleApplication field is set to `false`.

When an application is ineligible, the IneligibleMediationService is called to transform the CustomerApplication into an IneligibleApplication by using the data map you implemented earlier. The map transforms the data in the body of the service message objects when they are passed between the services. After transformation, the IneligibleApplication is sent to the RecordIneligibleApplication service for archiving by the flat file adapter.

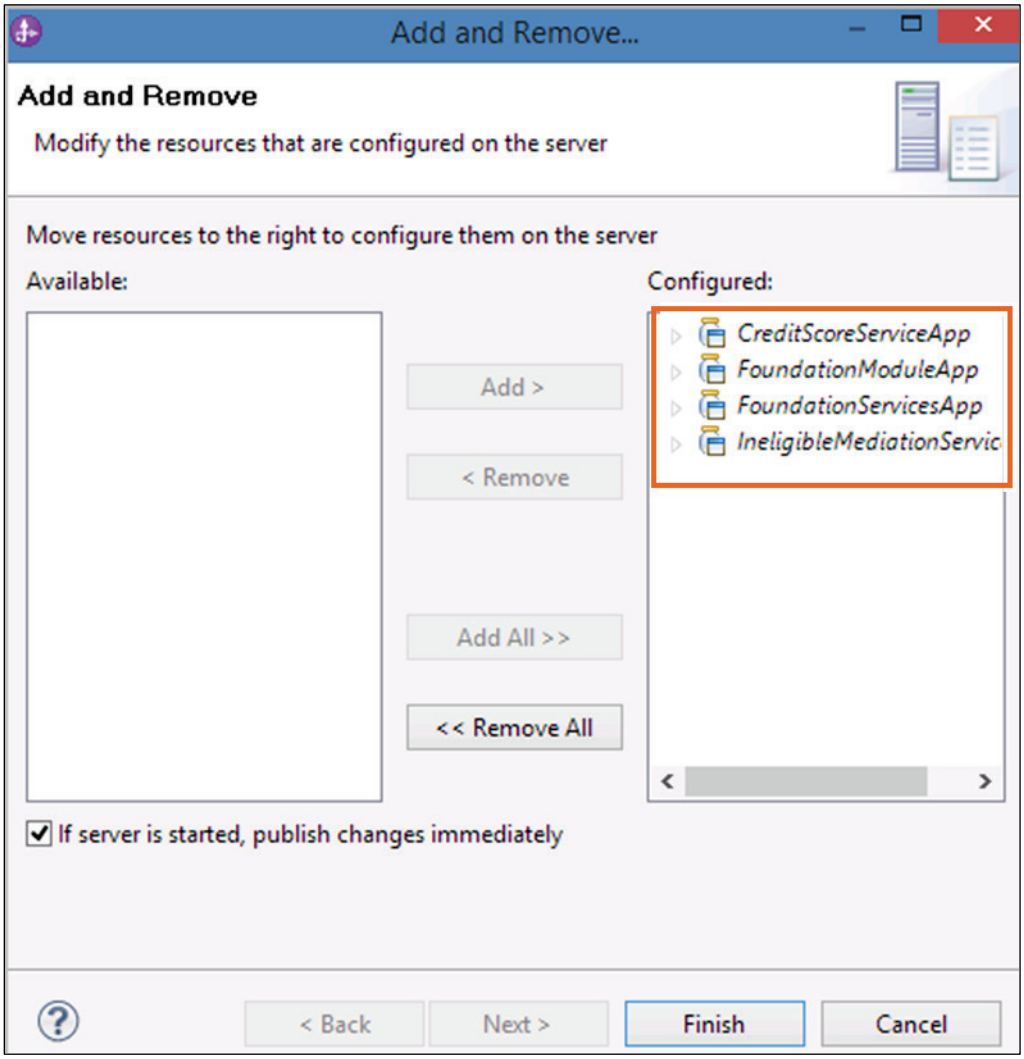
To test the IneligibleMediationService, you will clean the project. Before testing, it is a good idea to clean the project to make sure that no errors remain.

4. Click **Project > Clean** from the menu.
5. Verify that **Clean all projects** is selected and click **OK**.

   Wait a few minutes until the workspace is built before continuing to the next step.



2. Start the server (if it is not already running) and deploy IneligibleMediationServiceApp, FoundationServicesApp, CreditScoreServiceApp, and FoundationModuleApp.
   1. Start the Process Server (if not running) by double-clicking the desktop shortcut. Wait for the startup process to complete before continuing.
   2. In the **Servers** view, right-click **IBM Process Server v8.6 at localhost** and click **Add and Remove**.
   3. Click **Add All** to add the projects to the **Configured** list. You can also double-click projects individually to add them to the **Configured** projects list.



4. Click **Finish**.

   It takes a few minutes for the applications to publish and start. If you receive an error in publishing, then try restarting the individual applications.

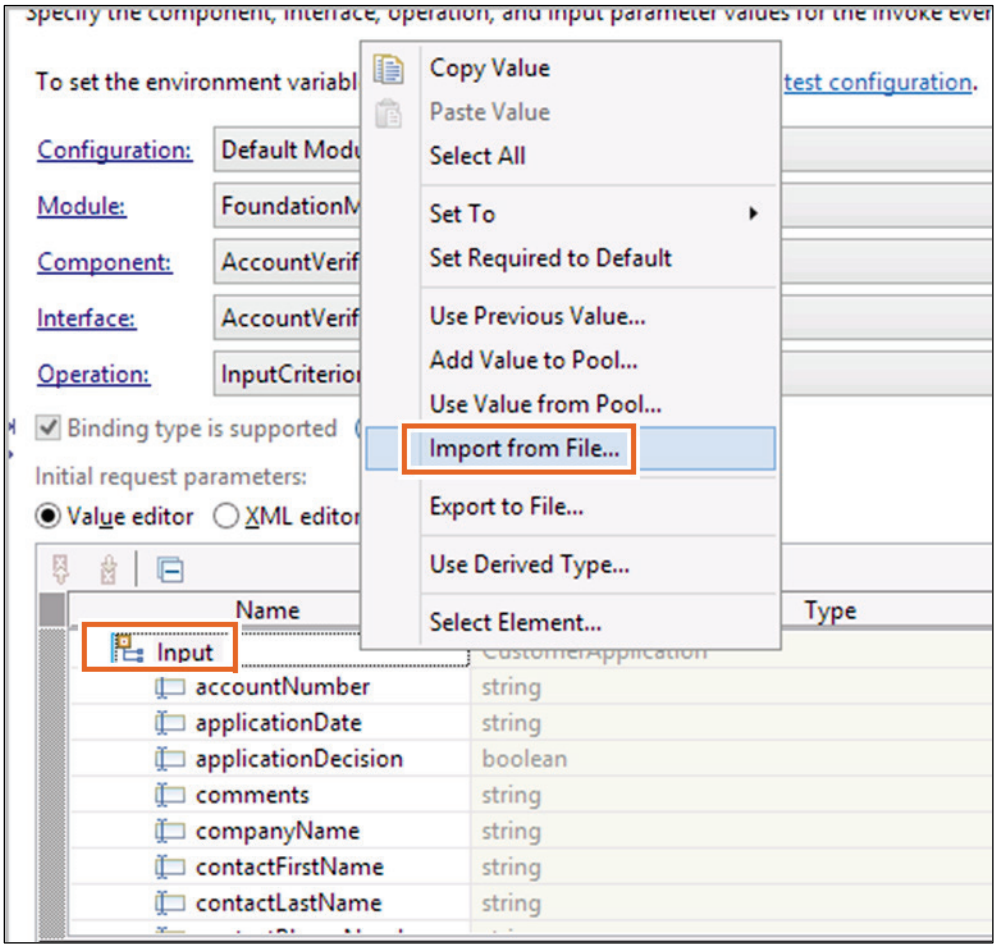
   When the applications are deployed and started, the messages Application started: CreditScoreServiceApp, Application started:IneligibleMediationServiceApp, Application started: FoundationServicesApp, and Application started: FoundationModuleApp are displayed in the **Server Logs** view.

5. If any module has a **Stopped** status, then right-click the module and click **Restart** from the menu. If prompted to do so, republish the module. Wait for the server status to change to **Started, Synchronized**. If the server has a status of **Started, Publishing**, then clicking the server refreshes the status to **Started, Synchronized**.

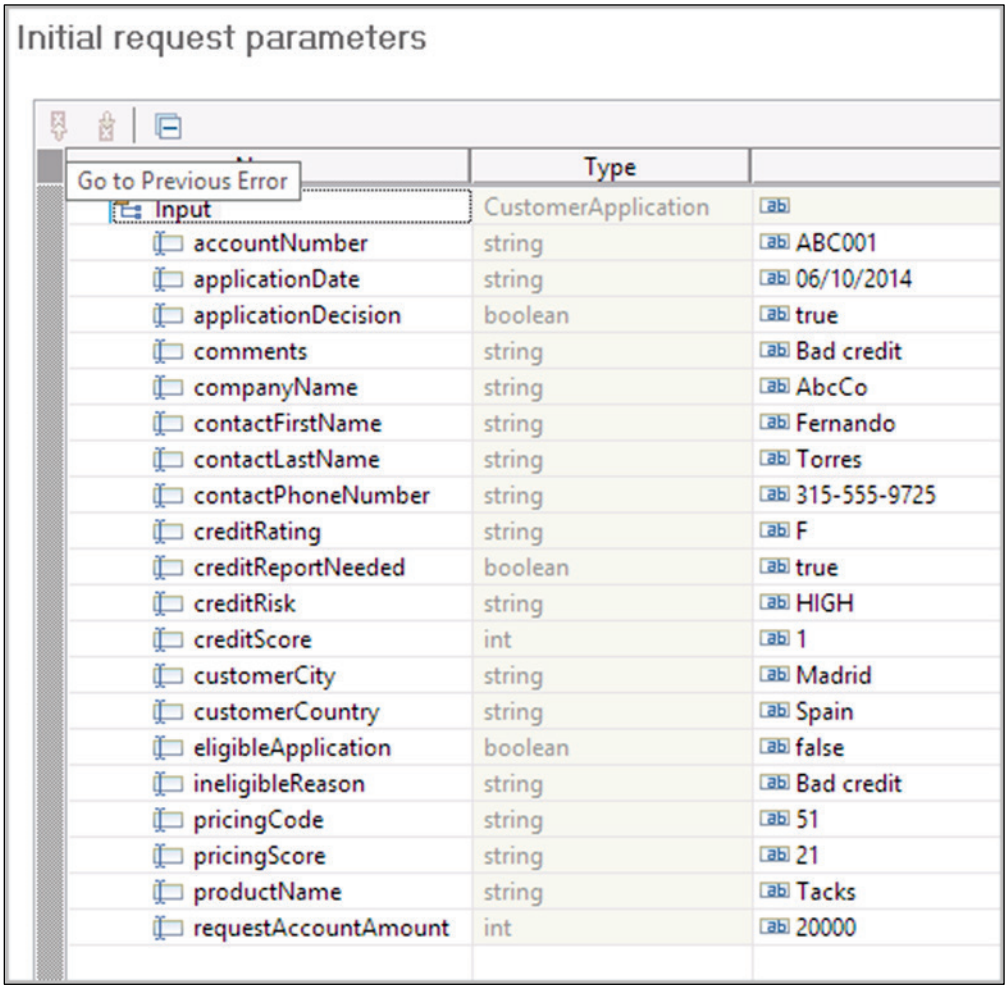
6. On the FoundationModule assembly diagram, right-click the **AccountVerificationExport** component and click **Test Component** from the menu.

3. Use C:\labfiles\Support Files\Ex10\EX10_Test_Data.xml to load the test data. Because you are testing the "ineligible" application path, you use the test data that is associated with company AbcCo. The test data for AbcCo automatically sets the eligibleApplication field to false.

   1. When the integrated test client opens in the **Initial request parameters** section, right-click **Input** and click **Import from File** from the menu.



2. Browse to `C:\labfiles\Support Files\Ex10\`, select **EX10_Test_Data.xml**, and click **Open** to populate the input parameters with the required test data.



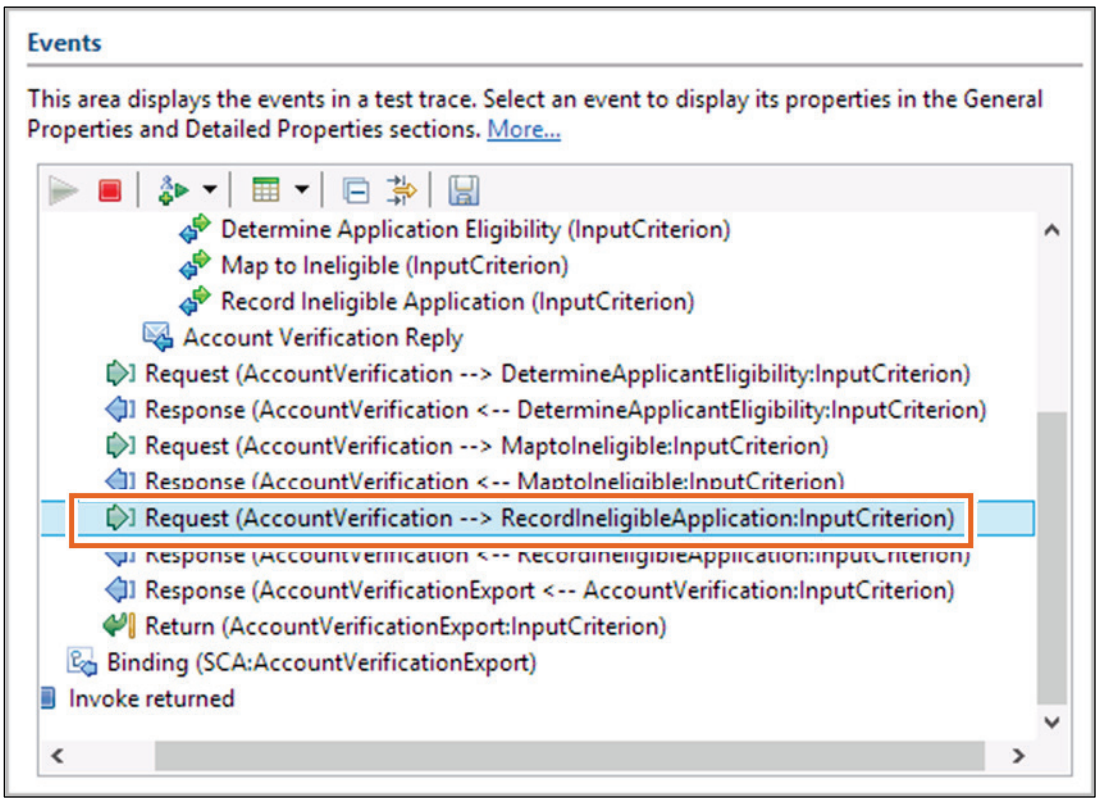
3. Alternatively, you can manually enter the following data (maximizing the initial request parameters window assists you with data entry).
   - accountNumber: `ABC001`
   - applicationDate: `06/10/2014`
   - applicationDecision: `true`
   - comments: `Bad credit`
   - companyName: `AbcCo`
   - contactFirstName: `Fernando`
   - contactLastName: `Torres`
   - contactPhoneNumber: `315-555-9725`
   - creditRating: `F`
   - creditReportNeeded: `true`
   - creditRisk: `HIGH`
   - creditScore: `1`
   - customerCity: `Madrid`
   - customerCountry: `Spain`
   - eligibleApplication: `false`
   - ineligibleReason: `Bad credit`
   - pricingCode: `51`
   - pricingScore: `21`
   - productName: `Tacks`
   - requestAccountAmount: `20000`
4. Click the **Continue** icon to run the test.
5. If the “Select a Deployment Location” dialog box is displayed, select **IBM Process Server v8.6 at localhost** and click **Finish**.

6. In the User Login dialog box, accept the default entries for **User ID** (`admin`) and **Password** (`web1sphere`) and click **OK**.

   Wait for the test to complete. When you receive the blue, square stop node in the **Events** window, the test is finished. Depending on resources, it might take a few minutes for the test to complete.

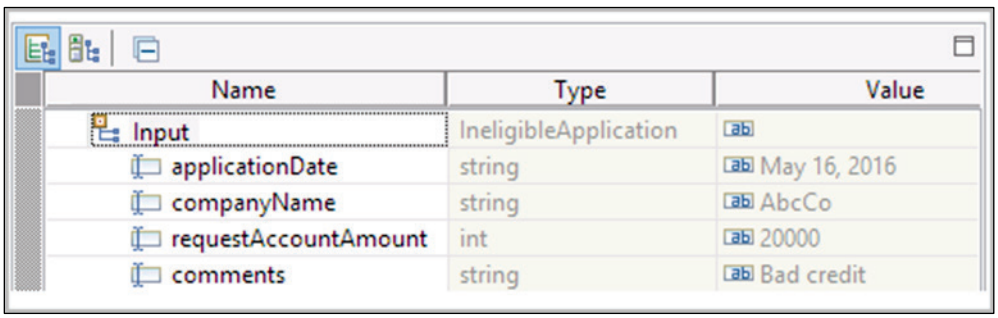
4. Examine the test trace.

1. Click the Request (AccountVerification > RecordIneligibleApplication:InputCriterion) event.

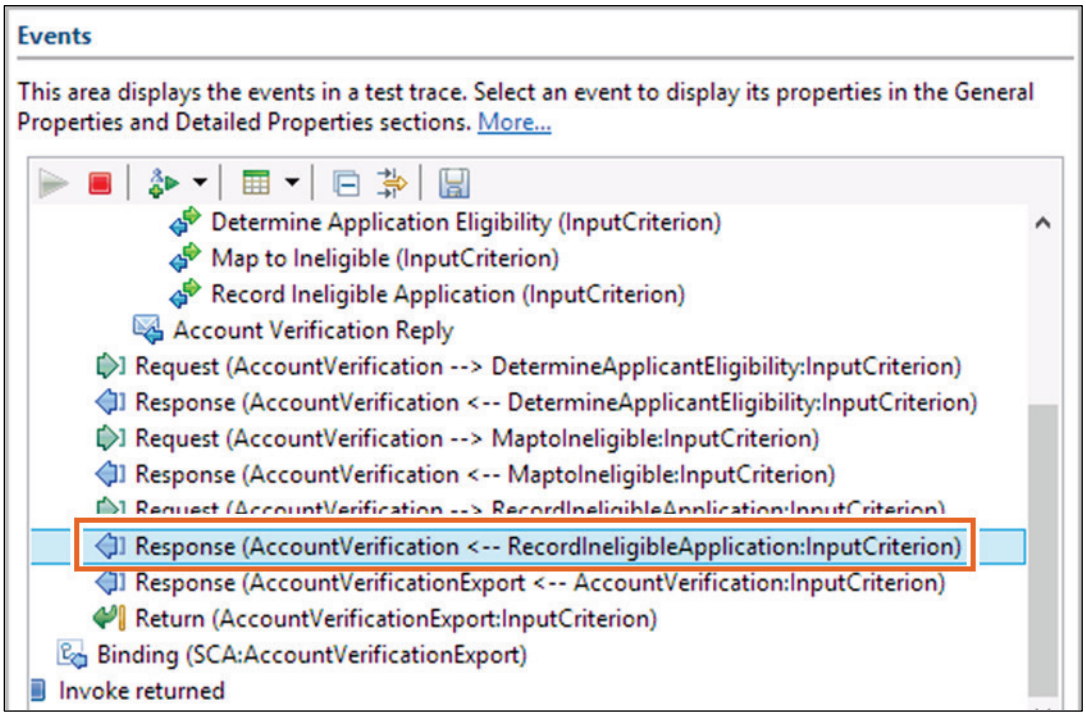


2. Examine the **Request parameters** section.

   The business object was correctly transformed from `CustomerApplication` to `IneligibleApplication`.

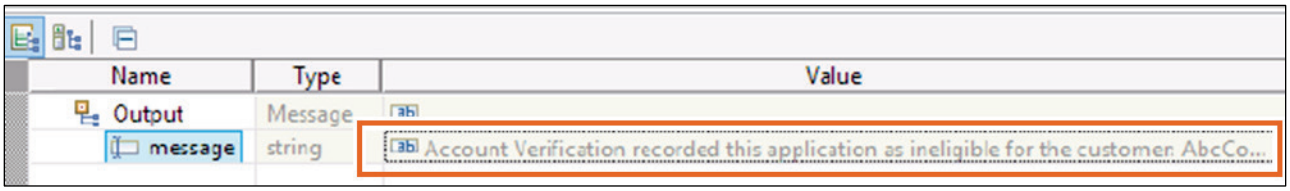


3. Select the **Response (AccountVerification <-- RecordIneligibleApplication:InputCriterion)** event.

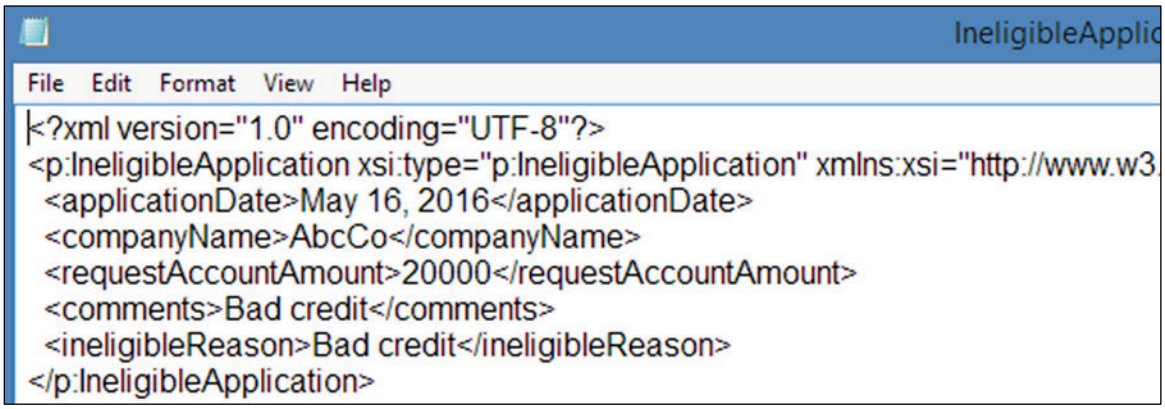


4. Examine the **Response parameters** section.

   Note the response message that indicates that the IBM WebSphere Adapter for Flat Files successfully archived the ineligible application. You can resize the window and expand the table to view the full text.



5. Open Windows Explorer and browse to `C:\IneligibleAppArchive\outdir`.
6. Open the file with the highest number at the end, for example, `IneligibleApplication.3.txt`. The test data corresponding to company `AbcCo` is displayed.



7. Close the ineligible application file and close Windows Explorer.
8. Close the **FoundationModule_test** tab and click **No** when you are prompted to save the test trace.

5. Remove the applications from the server and stop the server.

1. In the Servers view, right-click IBM Process Server v8.6 at localhost and click Add and Remove from the menu.
2. Click **Remove All** and click **Finish**.
3. Close IBM Integration Designer.

Results:
In this exercise, you defined an XML map and created a mediation module.

IBM Training

Mediation primitives

IBM Business Process Manager V8.6

IBM Training

Unit objectives

- Describe the role of mediation primitives in mediation flows
- Describe the prebuilt mediation primitives that are available for mediation flows

Unit objectives

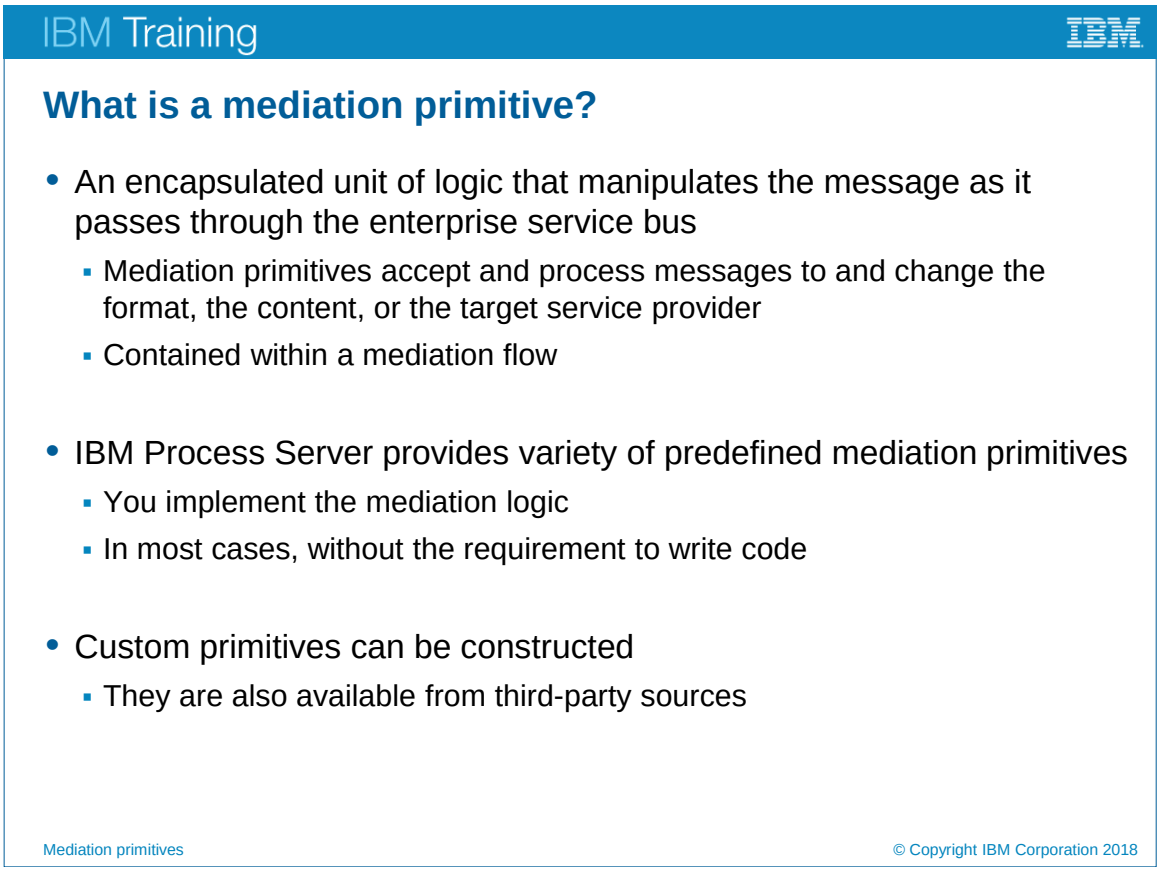
IBM Training

Topics

- Message transformation and enrichment primitives
- Flow control primitives
- Faults, tracing, and error handling primitives

Topics

Message transformation and enrichment primitives



What is a mediation primitive?

Mediation primitives are the core building blocks that are used to process the request and response messages in a flow. They are used to update, add to, and transform the SMO to control the flow for iteration or to make routing decisions, and do logging and event generation.

Built-in primitives do some predefined functions that are configurable by using properties. Custom Mediation primitives allow implementation of functions in Java.

Each mediation primitive presents a structure similar to this example. It has an input terminal, along with a set of output terminals (zero or more) and a fail terminal. Each terminal has a “type” defined with a WSDL message, and that type determines what you can “wire” the terminal to. Terminals must have compatible message types, or they cannot be connected to one another. The input and output terminals are the typical connection points for wiring mediation primitives into mediation flows.

The fail terminal also has a type; it has the same type as the input terminal. If an exception occurs during the execution of the primitive, the fail terminal is invoked, for example, if a database access exception occurs. In this case, the fail terminal propagates the original message, together with information about any exception that occurred, to the primitive to which it is connected. If no primitive is connected to the fail terminal, the flow fails (the processing of the message ends) with one of the following exceptions:

- MediationConfigurationException: The primitive can tell that a configuration problem or a transient problem exists, such as a resource availability issue.
- MediationBusinessException: A business error (for example, a key that is expected to be in a message is not found).
- MediationRuntimeException: Problems with setting up or establishing the flow in the runtime.

Failed flows also cause the current transaction to roll back.

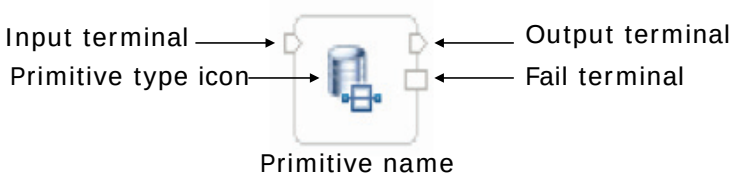
Mediation primitives operate on service message objects. XPath expressions can be used to access SMOs. Many configuration properties are XPath expressions. The XPath expression builder is used to set these properties. Expressions that are built with the expression builder are called custom XPath expressions. Several primitives have a configuration property that is called “root,” which represents the portion of the service message object that is being used.

Values for root include:

- `/`: The entire service message object
- `/body`: The body of the message (operation and parameter values)
- `/context`: The message context (transient context, correlation context, and failInfo)
- `/headers`: Protocol headers

IBM Training

Common features of mediation primitives



- Most mediation primitives share common attributes:
  - A default name, which is assigned when the primitive is moved to the drawing canvas
  - An input terminal, where the message is received
  - An output terminal, where the message is sent after the primitive successfully completes its actions
  - A fail terminal, where the original message and error information are sent when an exception occurs while the primitive is processing
- Some primitives have fewer or more terminals
- Some primitives have configurable properties that are associated

Mediation primitives

Common features of mediation primitives

IBM Training

Mediation primitives: Transformation (1 of 2)

Mediation primitive	Description
Business Object Map*	▪ Allows graphical creation of message transformations by reusable business object maps ▪ Use it to change the message content or the message type
Custom Mediation	Use Java code to implement custom mediation logic
Data Handler*	Converts an element of a message between a physical format and a logical structure
Database Lookup*	Modifies the content of a message by reading data from a user-supplied database

*See appendixes for details

Mediation primitives

Mediation primitives: Transformation

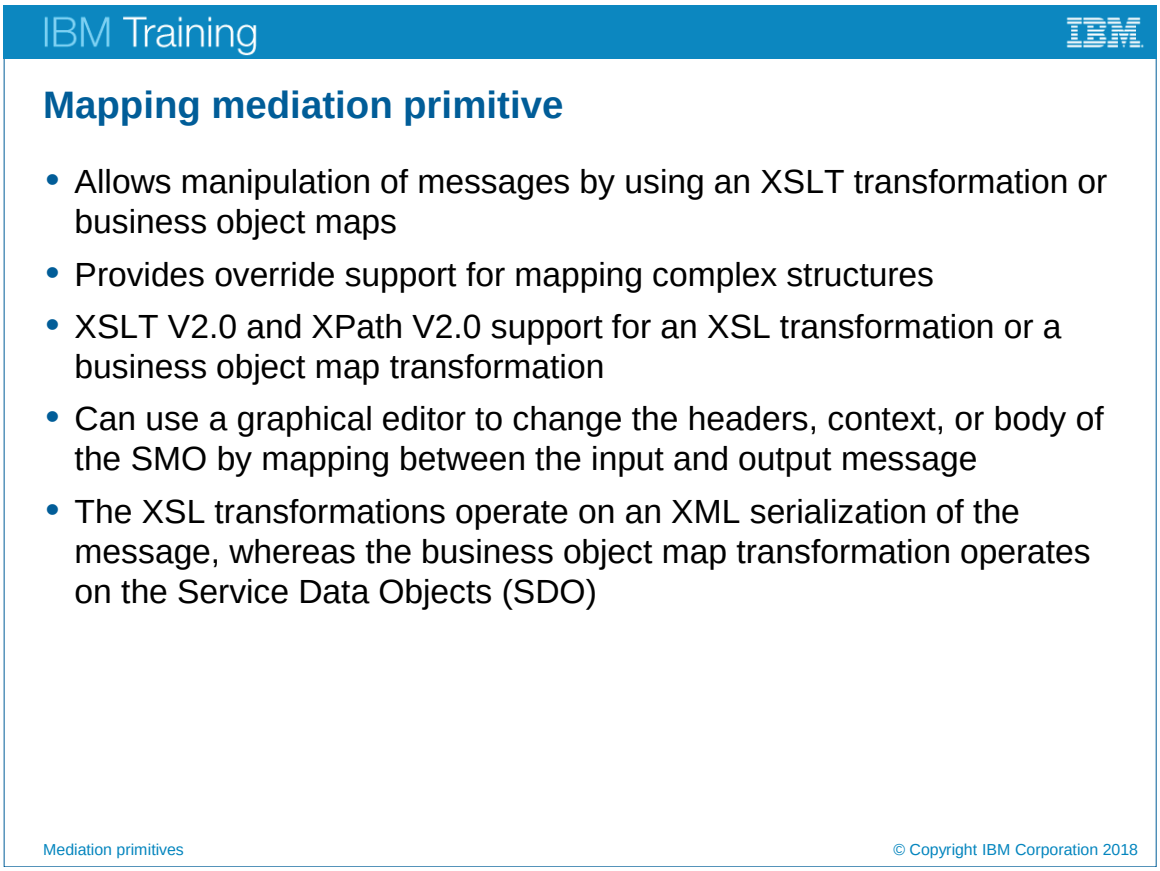
IBM Integration Designer provides a number of built-in mediation primitives that you can use to construct mediation flows. For the purposes of this course, the primitives are broken into three major groups: transformation, flow control, and tracing or error handling. Some primitives are included in the appendixes as reference material. Time constraints prevent coverage of all the primitives in this unit.

IBM Training

Mediation primitives: Transformation (2 of 2)

Mediation primitive	Description
Message Element Setter	Modifies the content (but not the type) of a message by adding, deleting, or changing message elements
Message header setters (four types)	▪ Creates, modifies, copies, or deletes message transport headers ▪ Specific header setter primitives exist for HTTP, JMS, WebSphere MQ, and SOAP messages
Set Message Type*	Overlays message fields with more detailed structures to allow easier manipulation of message content; treat weakly typed fields as strongly typed
Mapping	Allows modification of the message content by using Extensible Stylesheet Language (XSL) transformations or business object map transformations

*See appendixes for details



Mapping mediation primitive

You can use the Mapping mediation primitive to transform messages by using XSL transformations or business object maps. When you are integrating services, you often must transform data into a format that the receiving service can process. You can use the Mapping mediation primitive to transform one message type into a different message type.

You can use the Mapping mediation primitive to do the following tasks:

- Transform an input message type to a different output message type; for example, the mediation flow starts with one operation but ends with another operation, and the second operation has a different argument type
- Alter the content of a message without changing the message type
- Apply an existing XML map, or stylesheet, to transform a message

The Mapping mediation primitive can be useful if you want to:

- Manipulate data before or after the “Database Lookup” mediation primitive is invoked
- Copy the response from the Service Invoke mediation primitive into the shared context
- Use data in the shared context to create a message body after the Fan In mediation primitive

You can use either the Mapping mediation primitive or the Business Object Map mediation primitive to transform messages. The key difference is that the Business Object Map mediation primitive uses Service Data Objects (SDO) to do transformations on business objects. The Mapping mediation primitive can do transformations in XML, by using a stylesheet, business objects, and Service Data Objects.

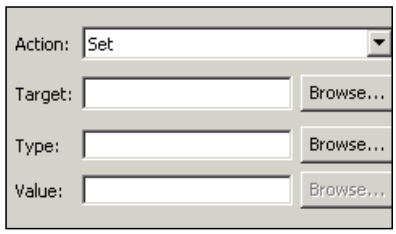
If you have existing XML maps, XSL stylesheets, or business object maps, you might be able to reuse them with the Mapping mediation primitive. Additionally, if you have existing business object maps, you might be able to reuse them with the Business Object Map mediation primitive. Some kinds of transformation are easier to do in XSL, and others by using a business object map.

IBM Training

Message Element Setter primitive

Modifies the body of an SMO by specifying:

- The element that must be modified (specified as an XPath expression)
- The operation that you want to perform on the target:
  - **Set**: Assigns a constant value to the target element
  - **Copy**: Copies a source element to a target element
  - **Append**: Copies from a source element to a new element instance, by appending to a repeating element in the output
  - **Delete**: Deletes an element instance
- The value that is going to be used, if a **copy** or **append** operation is selected (ensure that types are compatible, or runtime exception occurs)



- **Validate input** property: If set to “true,” and the input message is invalid (does not match its schema), a runtime exception occurs

You cannot use the Message Element Setter to change the ***type*** of a message, only the ***body*** of the message

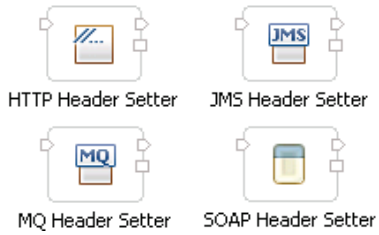
Mediation primitives

Message Element Setter primitive

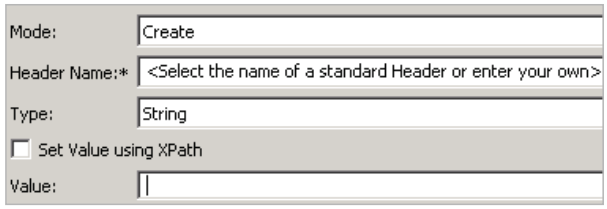
The Message Element Setter is used to modify the body of the SMO at the individual element level.

IBM Training

Message Header Setter primitives



- Four unique message header setter primitives:
  - HTTP Header Setter
  - JMS Header Setter
  - WebSphere MQ Header Setter
  - SOAP Header Setter
- Modifies the transport protocol header of an SMO message by specifying:
  - Type (“mode”) of operation you want to do (create, copy, modify, or delete)
  - The name of the header to be modified
  - The value that is used if a create, copy, or modify operation is requested
  - A flag that indicates whether the value is an XPath expression or a literal



- **Validate input** property: If set to “true,” and the input message is invalid (does not match its schema), a runtime exception occurs

You cannot change the ***body*** of a message that uses the message header setter primitives, only the specified transport protocol headers

Mediation primitives

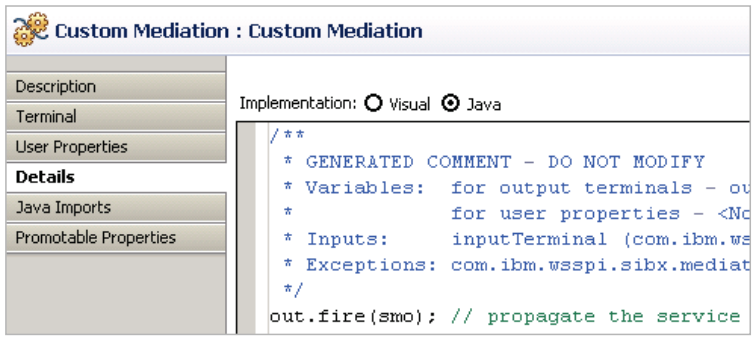
Message Header Setter primitives

Four message header setter primitives are available, one for each of the transport protocols that are most often used in the IBM Process Server environment: JMS, HTTP, WebSphere MQ, and SOAP. This group of primitives is used to modify the transport protocol header, for example, changing the message destination in the JMS or WebSphere MQ headers.

IBM Training

Custom Mediation primitive

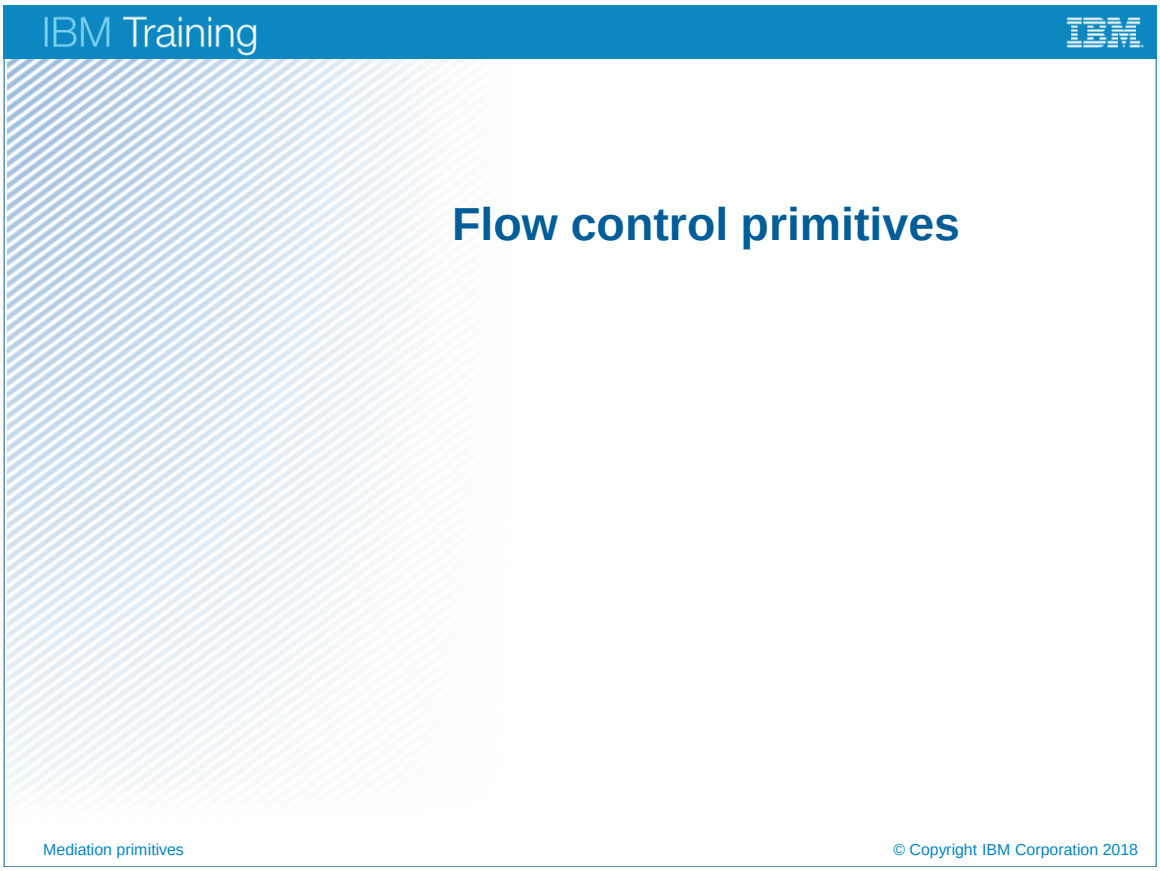
- Allows implementation of custom mediation logic in mediation flows by using Java snippets or visual snippets
- Useful when no built-in primitive provides the needed function
- Can define more input and output terminals and properties on the primitive at development time for increased flexibility
- Must explicitly send the message through the output terminal by using the `out.<terminal name>` notation
- Must explicitly define the message types on the terminals
- When a Custom Mediation primitive is created on the canvas, the initial "skeleton" code is created for you



Mediation primitives

Custom Mediation primitive

The Custom Mediation primitive allows writing your own Java code to manipulate the SMO.



Flow control primitives

IBM Training

Mediation primitives: Routing (1 of 2)

- These primitives control service invocations inside the mediation flow and message routing outside the mediation flow

Mediation primitive	Description
Message Filter	Routes messages within a mediation flow based on the message content
Service Invoke	Calls a service from within a mediation flow (instead of waiting until the end of the mediation flow and then implementing the callout mechanism)
Type Filter	Routes messages within a mediation flow based on the message type

Mediation primitives

Mediation primitives: Routing

This group of built-in mediation primitives is used to route the flow of messages within a mediation, or to dynamically control how services are invoked. As in the previous topic, some primitives are included in the appendixes as reference material. Time constraints prevent coverage of all the primitives in this unit.

IBM Training

Mediation primitives: Routing (2 of 2)

- These primitives control routing within the mediation flow

Mediation primitive	Description
Fan In*	Aggregates multiple messages (created by a Fan Out primitive) based on a decision point, and then outputs a single message
Fan Out*	Splits a message that contains repeating elements into multiple messages, or sends the same message more than one time to implement message aggregation
Flow Order*	Specifies the order in which the branches of a flow are run

*See appendixes for details

IBM Training

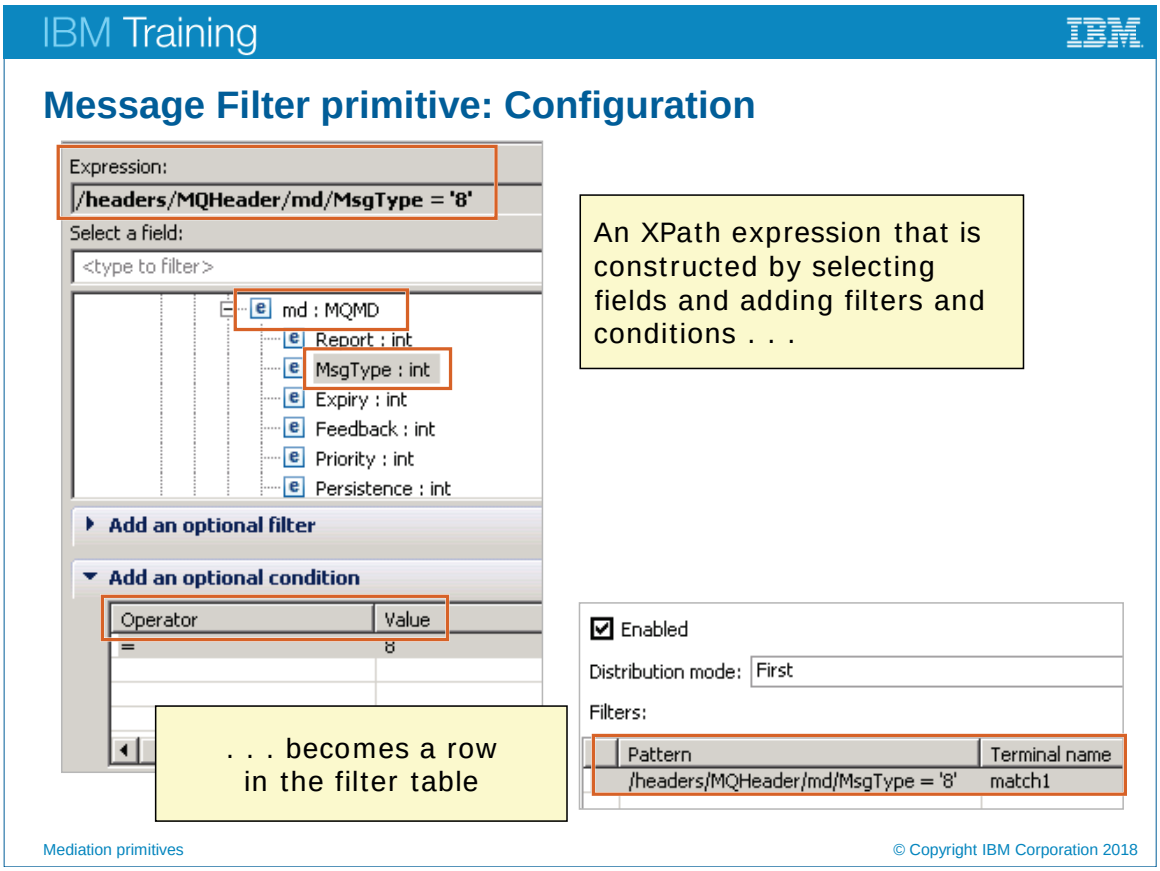
Message Filter primitive

- Routes a message within a mediation flow based on the message content
- Acts like an “if . . goto” statement to send an incoming message through one or more paths if the filter criteria are met
- Any number of output terminals can be defined
- Filter criteria are XPath expressions in a table
- If no criteria match, then the message is propagated through the **default** terminal
- Usage:
  - Verify that the contents of a message meet conditions, such as the required fields are present
  - If the conditions are not met, invoke the error handler subflow or Fail primitive to stop processing
  - Route the message based on its contents (the value in the field indicates that service provider A is invoked; other values indicate that service provider B is invoked)
  - Conditionally bypass steps in a mediation based on the message content

Mediation primitives

Message Filter primitive

Use care when wiring the message filter outbound terminals. If you do not wire a flow to a terminal and the message is propagated to that unwired terminal at run time, the message is lost without warning or exception. This attention is especially important with the **default** terminal, which receives a message that does not match any of the XPath expressions in the table.



Message Filter primitive: Configuration

To configure the Message Filter primitive:

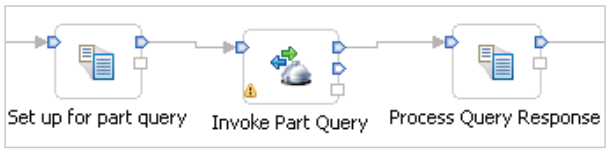
1. You construct a table of XPath expressions to evaluate the contents of elements or variables. You can reference any content in the SMO, not just that of the incoming message. The XPath expression builder helps you construct the expression.
2. If that condition is met, you create a terminal (or use an existing one) to receive the incoming message.
3. Repeat those steps for all conditions that you want to test.
4. Set the distribution mode: “first” sends the message only on the first true condition that is found, as the table of expressions is scanned at run time; “all” sends the message to all terminals where the condition is true. (This distribution mode implies that the order in which expressions are listed in the table can be significant.)
5. Set the enabled flag. You can disable a Message Filter without removing the primitive from the mediation flow. This property can also be promoted, so it can be changed administratively without altering the mediation flow.

At run time, the table of expressions is scanned in sequential order. If an XPath expression evaluates to true, the incoming message is sent out through the terminal that is listed in the table for that expression. Depending on the value of the distribution mode property, the table scan either stops or continues.

IBM Training

Service Invoke primitive: Overview

- Similar to the callout node, but it can be used anywhere in a request flow
- Sends a message to (invokes) a requested service and operation
- Has **in** and **out** terminals for inbound and outbound messages
- Has a **timeout** terminal that is fired when the timeout threshold is exceeded, like a callout node
- Has a <**fault message name**> terminal for each modeled fault
  - When a modeled fault occurs, it propagates the modeled message that is returned from the invocation



A sample flow that invokes a service

Mediation primitives

Service Invoke primitive: Overview

By using the Service Invoke primitive, it is not necessary to wait until the end of the mediation flow before invoking a service. In every other aspect, it behaves identically to the callout node.

IBM Training

Service Invoke primitive compared to callout node

- Service Invoke is embedded in a mediation flow such as callout, callout response, and callout fault
- Service Invoke mediation primitive versus callout:
  - Service Invoke mediation primitive does not switch from the request flow to the response flow; callout does
  - Service Invoke mediation primitive does not modify either the transient context or the correlation context; callout does
- When to use which:
  - To mediate a message (without calling an intermediate service) and call a service provider, use a callout
  - Use Service Invoke to call a service and return a result in a request flow without invoking the response flow; multiple Service Invoke primitives can exist in the same flow
  - To call an intermediate service, use Service Invoke and callout to invoke the final service
  - To invoke multiple intermediate services, use multiple Service Invoke primitives; you might not need the callout node
  - Interface and reference must not be wired when you use a Service Invoke to call a service and immediately return the response to the user
  - No response flow is needed in this case

Service Invoke primitive compared to callout node

IBM Training

Type Filter primitive

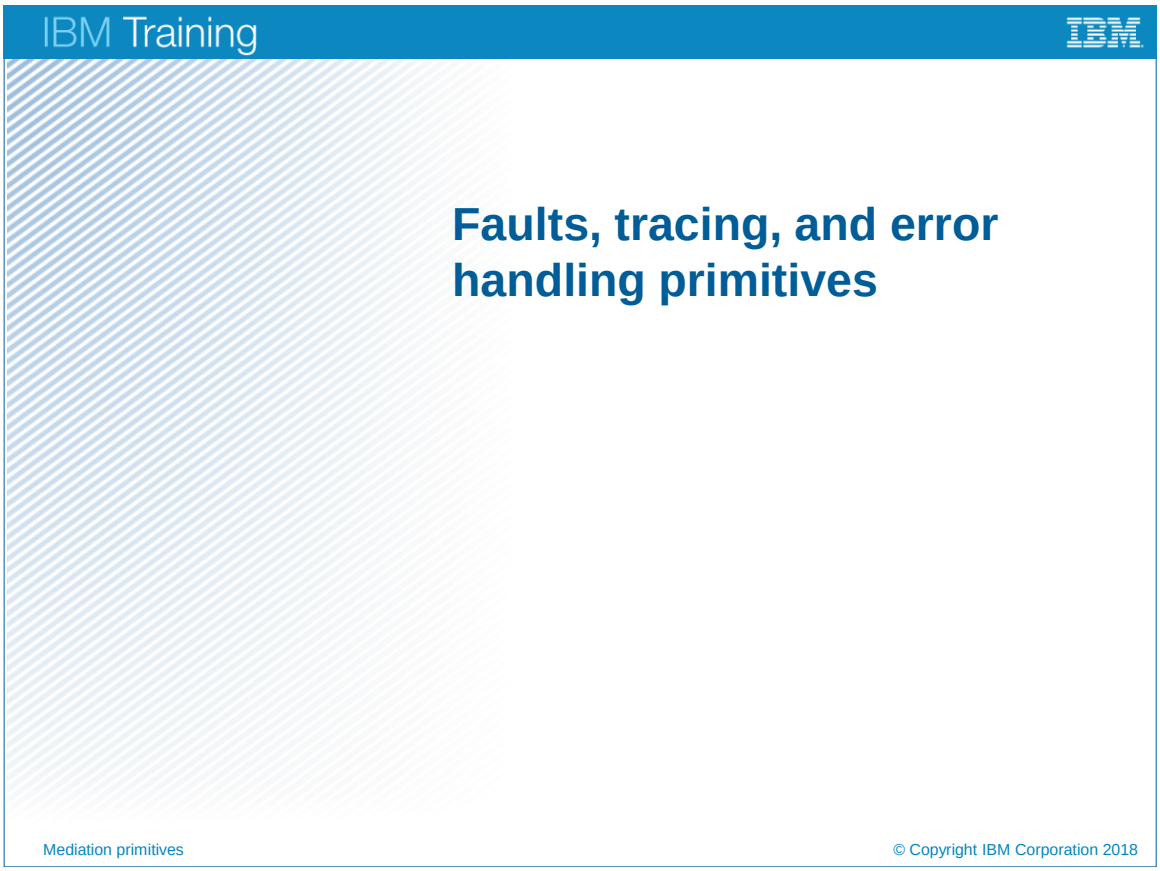
- Routes a message within a mediation flow based on the message type
- Acts like an “if . . goto” statement to send a message through a specific path if the filter criterion is met
- Any number of output terminals can be defined at development time
- Filter criteria are XPath expressions that you enter in a table at development time
- If no match occurs, the message is propagated through the *default* terminal

Mediation primitives

Type Filter primitive

The Type Filter is similar to the Message Filter, except that it evaluates the type of an element, rather than the content of an element.

Unlike the Message Filter primitive, the Type Filter primitive has no distribution mode property. At run time, the first expression that matches during the scan of the table propagates the inbound message to the corresponding terminal. There is no means to match multiple rows in the expression table. Thus, the order in which the expressions are listed can be important.

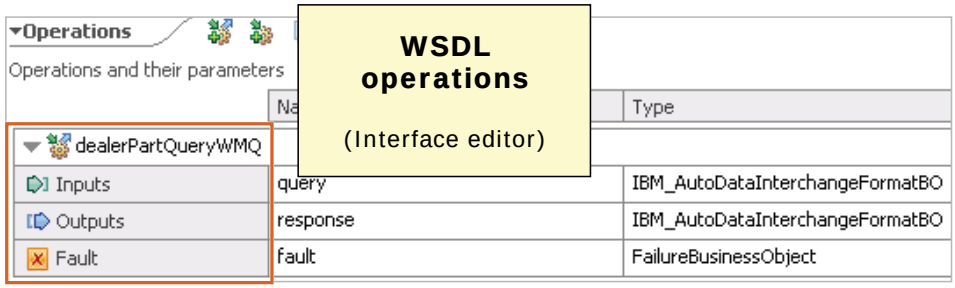


Faults, tracing, and error handling primitives

IBM Training

Faults

- Three types of WSDL operations: input, output, and fault (can be multiple faults)
- For each fault that is defined in WSDL, a corresponding terminal is created on the input fault node
  - If you define the fault operation in WSDL (the interface editor) for a request flow, an **input fault** node is created
  - Corresponding nodes are created on the response flow for handling faults



Mediation primitives

Faults

You can define three types of WSDL operations on an interface. You see the interface editor here, displaying those operations. Whether you define a fault operation determines whether more nodes are added to the mediation to support WSDL error handling.

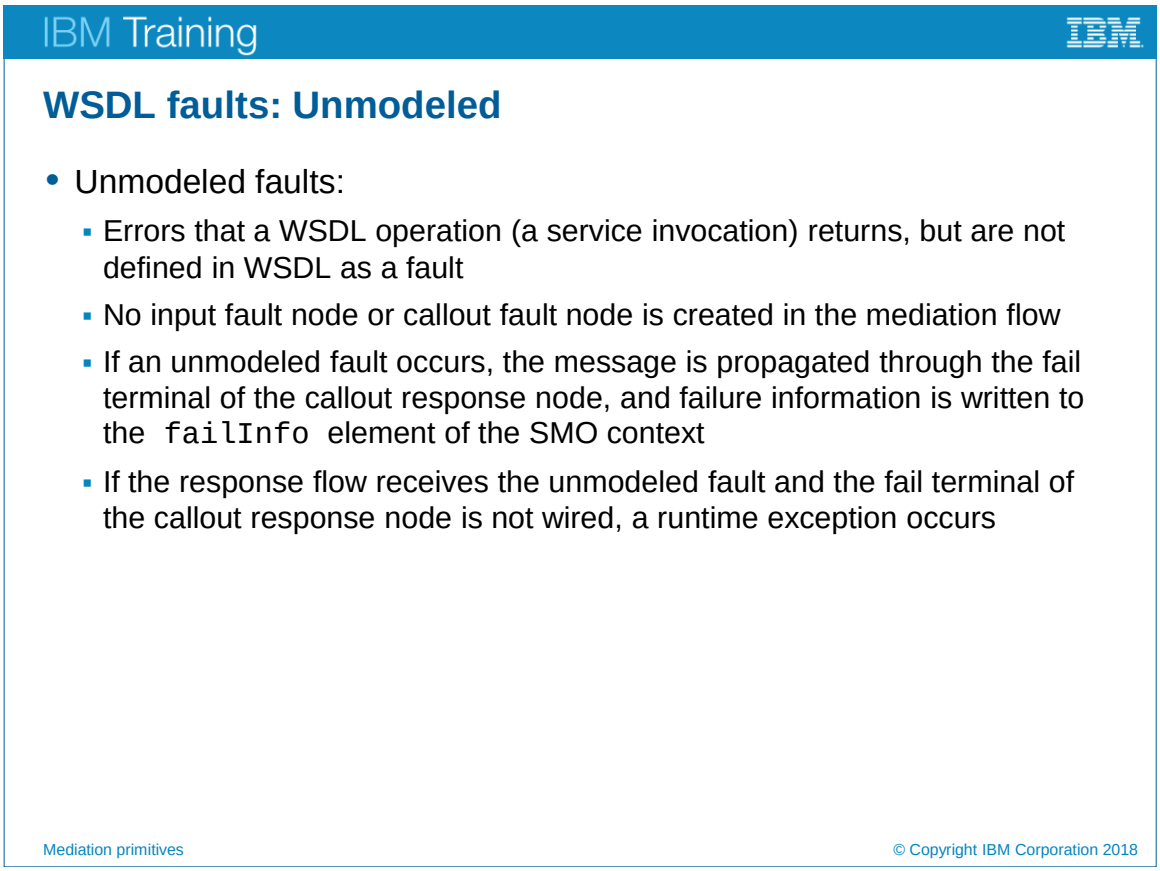
IBM Training

WSDL faults: Modeled

- Two types of WSDL faults: modeled and unmodeled
- **Request flow**: Modeled faults
  - Begins with an input node and (normally) ends with a callout node
  - If a fault is defined in WSDL for a request flow, an **input fault** node is created
  - If an error occurs during request processing, wire a branch of the request flow to the input fault node to return an error message to the caller
  - Do this wiring instead of invoking a callout
  - For each fault defined in WSDL, a corresponding terminal on the input fault node is created
- **Response flow**: Modeled faults
  - Begins with the callout response node and (normally) ends with the input response node
  - If a fault is defined in WSDL for a response flow, the **callout fault** node is created to handle errors that the called service generates
  - For each fault defined in WSDL, the corresponding terminal on the callout fault node is created
  - Can wire the callout fault node to the input fault node to send errors back to the client that the called service detects

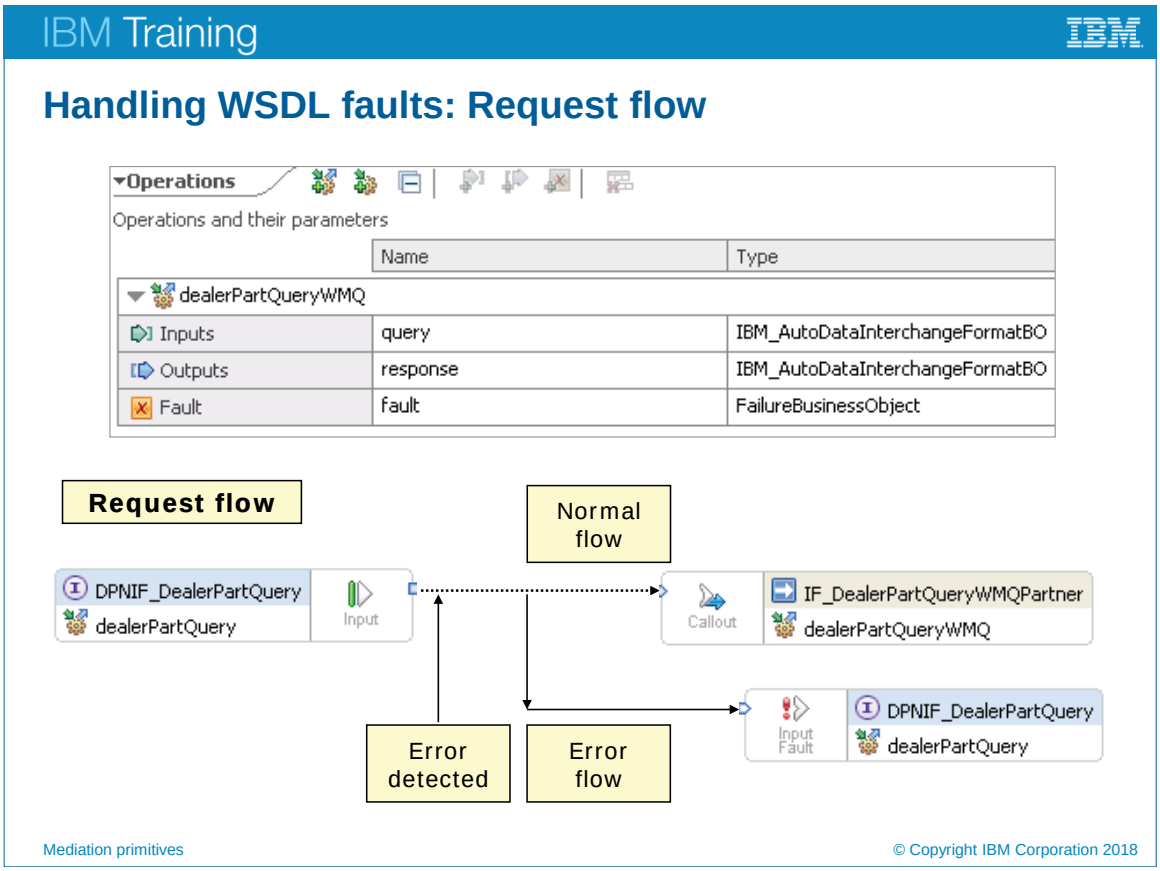
WSDL faults: Modeled

The two types of WSDL faults are modeled and unmodeled. A modeled fault is one for which a fault is defined in the WSDL interface so that the mediation can have a defined reaction if that fault arises during the service invocation.



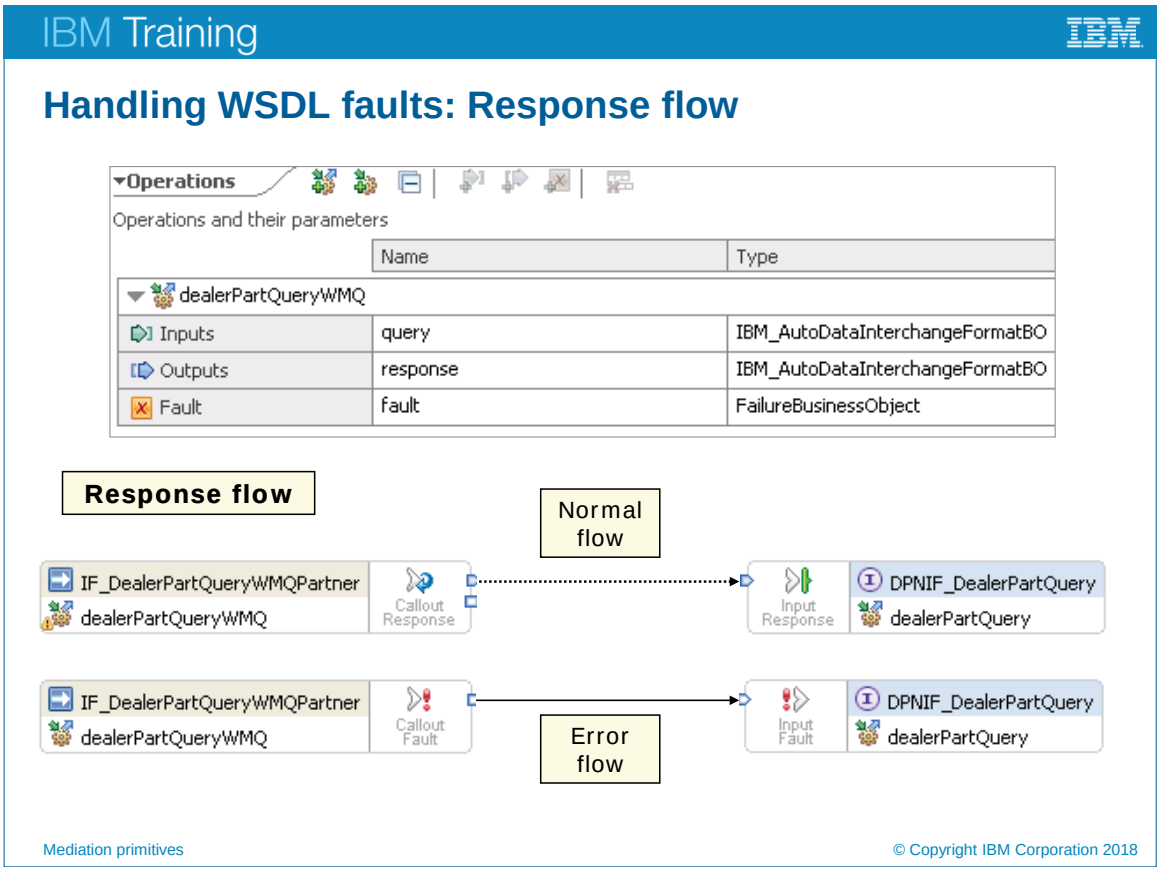
WSDL faults: Unmodeled

An unmodeled WSDL fault is a fault that is returned from a service invocation, but for which there is no means provided in the mediation flow to handle it. There is no means to handle it because the fault was not defined in the WSDL interface. How IBM Process Server responds to unmodeled faults depends on how the mediation is wired.



Handling WSDL faults: Request flow

In this example of a request flow, the WSDL interface is defined to include a fault operation. During the execution of the request flow, a determination is made that indicates that the message data has a problem. Rather than send the invalid data to the callout node, a primitive (such as the Message Filter) instead routes the message to the input fault node. The input fault node returns the message to the caller without doing the callout operation at the end of the request flow.



Handling WSDL faults: Response flow

In the normal situation in a response flow that follows a Service Invoke or callout operation, the response message flows to the input response node. However, in the situation where the service invocation returns a fault, the message travels from the callout fault node to the input fault node. Again, the assumption is that the fault is a modeled fault; that is, you provide for the fault in the WSDL interface. If an unmodeled fault occurs, the behavior is as described in the “WSDL faults: Unmodeled” slide (two visuals previously), which uses the fail terminal of the Service Invoke, or a runtime exception occurs.

IBM Training

Mediation primitives: Error handling, debugging, and event recording

Mediation primitive	Description
Event Emitter	Sends a monitoring event (in the Common Base Event format) from within a mediation flow to a Common Event Infrastructure server
Fail	Stops processing in a mediation flow and generates an exception
In*	Is used as an entry point in a subflow
Out*	Is used as an exit point in a subflow
Message Logger	Stores messages in a relational database or other medium
Message Validator*	Validates all or part of a message against its schema
Stop	Stops processing in a mediation flow, without generating an exception
Trace	Writes trace messages to a server log or to log files

*See appendixes for details

Mediation primitives

Mediation primitives: Error handling, debugging, and event recording

These built-in mediation primitives are used for error handling in mediation flows and debugging and recording or logging of events that occur at run time. Some primitives are included in the appendixes as reference material. Time constraints prevent coverage of all the primitives in this unit.

IBM Training

Message Logger primitive

- Stores a message in a relational database table
- Can also write a message to another medium by using the custom log facility
- Saves a selectable amount of the message content plus identifying information (time stamp, message identifier, primitive and module names, and other information)
- Messages can be logged for later review, post-processing, auditing, or whatever purposes you require
- Original inbound message passes through without alteration

Mediation primitives

Message Logger primitive

The following list is an overview of the Message Logger properties:

- **Enabled**: Determines whether the logger does its function; it allows logging to be disabled without removing the primitive. The administrator promotes or sets this property.
- **Root**: XPath expression that describes the message to log. The message to log is converted to XML from the point that Root specifies.
- **Transaction mode**: Determines whether Message Logger participates in a mediation transaction or acts independently.
- **Logging type**: The destination type for the logging event: database or custom (uses the custom logging facility to write to a flat file, for example).
- **Data source name**: The JNDI name of the data source that defines where the data is logged. The default value points to a database that is named CommonDB.
- **Handler, Formatter, Filter, Literal, Level**: Implementation classes that are used when custom logging is enabled.

By default, messages are written to a data source that is named CommonDB, to a table that is named MSGLOG.

By using the default value for the Literal property, the call to `MessageFormat.format(<LogRecord>.getMessage(), <LogRecord>.getParameters())` in the default Formatter implementation class means the following values:

- `{0}` would then be replaced with the Time Stamp value: `logMessageParameters[0]`
- `{1}` would then be replaced with the Message ID value: `logMessageParameters[1]`
- `{2}` would then be replaced with the Mediation Name value: `logMessageParameters[2]`
- `{3}` would then be replaced with the Module Name value: `logMessageParameters[3]`
- `{4}` would then be replaced with the Message value: `logMessageParameters[4]`
- `{5}` would then be replaced with the version value: `logMessageParameters[5]`

IBM Training

Trace primitive

- Writes a message to the server log, user trace log file, or another file
- Can be used for debugging or tracing a mediation flow
- Saves a selectable amount of the message content plus identifying information (time stamp, message identifier, primitive and module names, and other information)
- Original inbound message passes through without alteration

Trace primitive

IBM Training

Stop primitive

- Stops the execution of a mediation flow
- Consumes the incoming message silently, and then stops the flow
- Contains only an **in** terminal, no **out** or **fail** terminals
- Has no configurable properties
- If wired to a normal output terminal, the behavior is the same as if the output terminal were unwired (the message is lost, silently)
- If wired to a fail terminal, the exception from the fail terminal is consumed silently, rather than propagated

Stop primitive

IBM Training

Fail primitive

- Stops the execution of a mediation flow
- Consumes the incoming message, raises a `FailFlowException`, and then stops the flow
- You can specify an error message, and define part or all of the SMO to include with the exception (the default SMO content that is listed is `/context/failInfo`)
- Uses the same message format (with substitution variables) as the Message Logger primitive

Mediation primitives

Fail primitive

The Fail primitive uses the same message format and content as the Message Logger primitive, including substitution variables `{0}` through `{5}`:

- `{0}` would then be replaced with the Time Stamp value: `logMessageParameters[0]`
- `{1}` would then be replaced with the Message ID value: `logMessageParameters[1]`
- `{2}` would then be replaced with the Mediation Name value: `logMessageParameters[2]`
- `{3}` would then be replaced with the Module Name value: `logMessageParameters[3]`
- `{4}` would then be replaced with the Message value: `logMessageParameters[4]`
- `{5}` would then be replaced with the version value: `logMessageParameters[5]`

If you supply the optional error message in the Properties view of the Fail primitive, it is substituted for message `{4}`.

IBM Training

Unit summary

- Describe the role of mediation primitives in mediation flows
- Describe the prebuilt mediation primitives that are available for mediation flows

Unit summary

IBM Training

Checkpoint questions

1. True or False: A Service Invoke mediation primitive allows a service to be invoked from within a request or response flow, rather than waiting to reach a callout node.
2. True or False: It is possible for more than one condition in a Message Filter primitive expression table to match.
3. True or False: Using the Event Emitter to log every incoming message for audit purposes would be considered a good practice.
4. True or False: You can configure the Stop primitive to display a user-defined message when it stops the mediation flow.

Checkpoint questions

IBM Training

Checkpoint answers

1. True.
2. True, only if the distribution mode property is set to “all.”
3. False: An Event Emitter should not normally be in the main flow of a mediation because of the message traffic (and related potential effect on performance).
4. False: The Stop primitive has no configurable properties, but you can set the Fail primitive to display a user-defined message.

Checkpoint answers

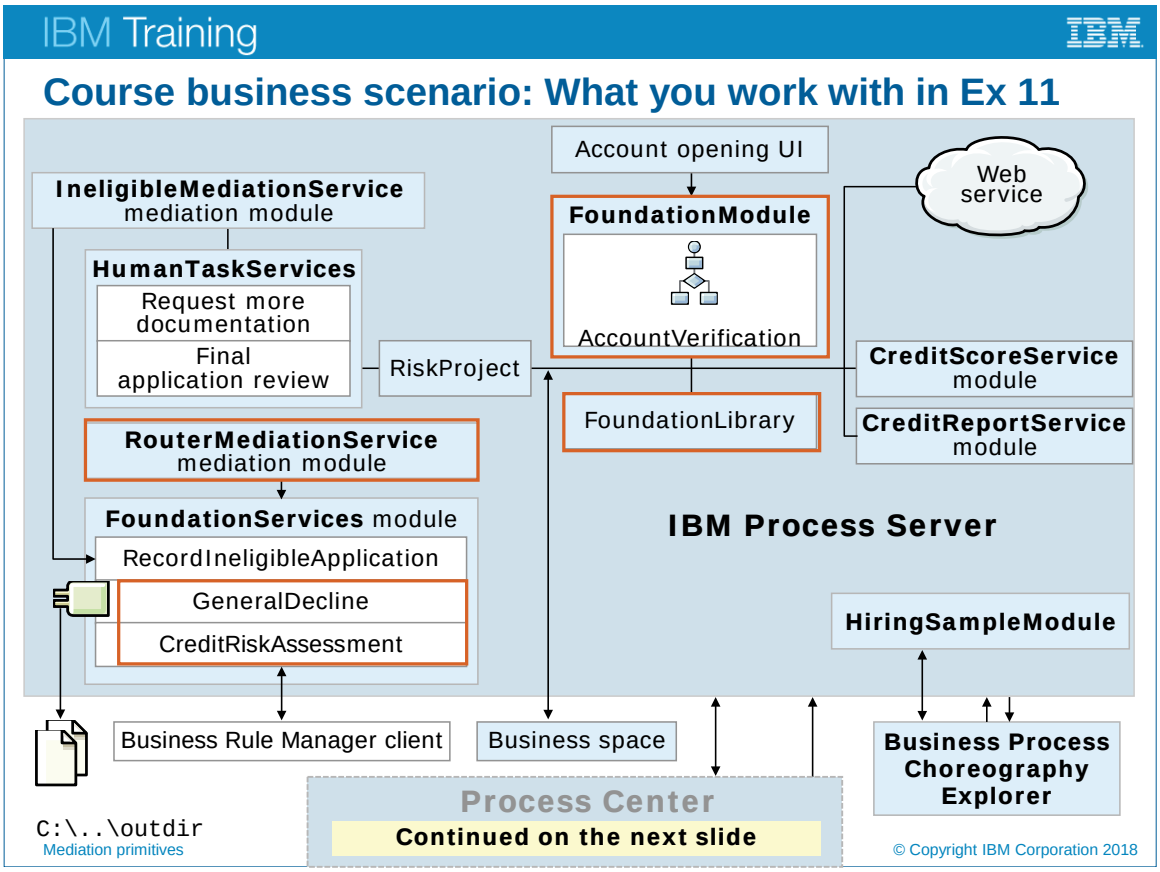
IBM Training

Exercise 11: Creating mediation services, part II

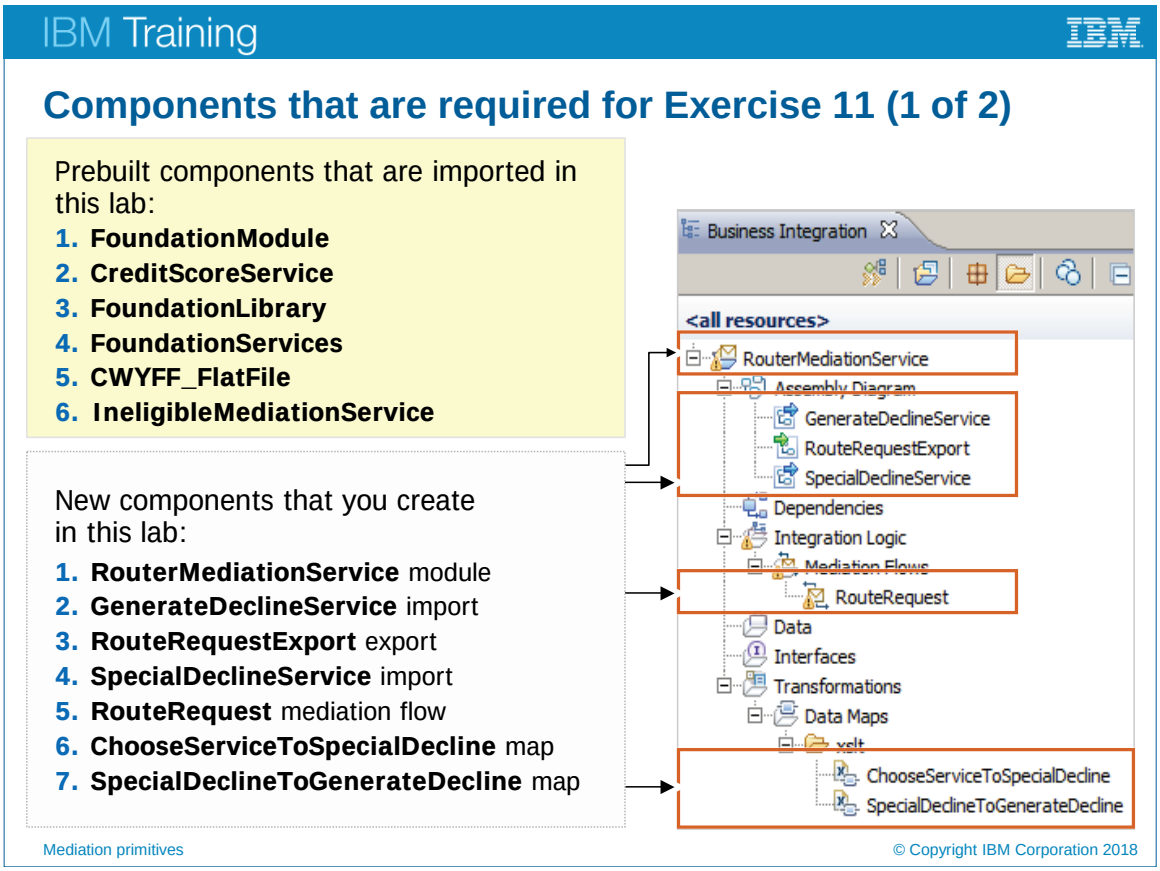
After you complete this exercise, you should be able to:

- Create a mediation module that contains a Message Filter mediation primitive and a Mapping primitive
- Define an XML data map
- Test a mediation module that contains a Message Filter mediation primitive and a Mapping primitive

Exercise 11: Creating mediation services, part II



Course business scenario: What you work with in Ex 11



Components that are required for Exercise 11

If applicationDecision is set to false during FinalApplicationReview (the application is declined) and the customer's creditRisk is HIGH, the application is routed through the "generate decline" component. If applicationDecision is set to false during FinalApplicationReview and the customer's creditRisk is MED (short for medium), the application is routed through the "special decline" component.

In this portion of the exercise, you implement the mediation flow for the RouteRequest component. The RouteRequest flow component contains the mediation logic that routes the application to the appropriate decline service.

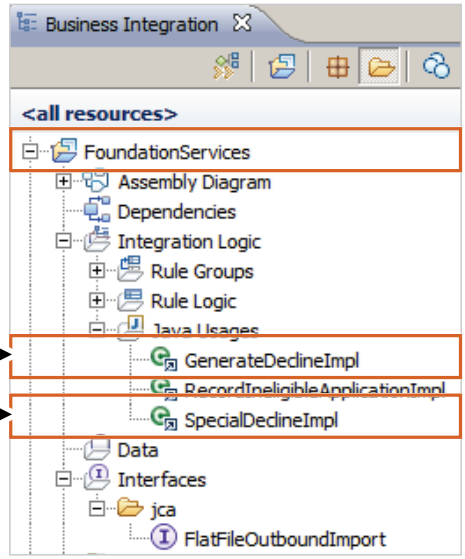
The RouteRequest mediation flow consists of both a request flow and a response flow. In the flow, the CustomerApplication is routed to the appropriate decline service by a router mediation primitive. After processing, the response from the decline service is sent back to the AccountVerification process.

IBM Training

Components that are required for Exercise 11 (2 of 2)

Java components that you add in this lab:

1. **GenerateDeclineImpl**
   Java implementation
2. **SpecialDeclineImpl**
   Java implementation



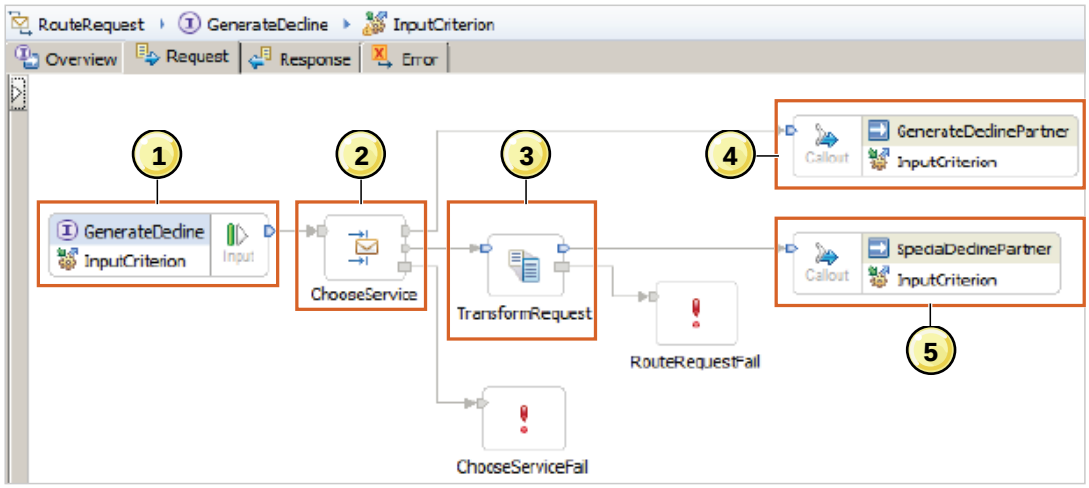
Mediation primitives

If the employee approves the application, the process completes successfully. However, if the application is declined, it is routed to one of two possible services. If the application is declined and the creditRisk is HIGH, the generate decline service is invoked. It is the **GenerateDeclineImpl** implementation code that you add in this exercise. The code returns a Message business object that contains the message: "`Account for customer <company name> was declined and the credit risk was <credit risk>.`"

If the application is declined and the creditRisk is MED, the special decline service is invoked. It is the **SpecialDeclineImpl** implementation code that you add in this exercise. The "special decline" service is called when the application is declined during FinalApplicationReview, and the creditRisk is MED. The code in SpecialDecline returns a Message business object that contains the message: "`Account for customer <company name> was routed through special decline because the credit risk was <credit risk>.`"

IBM Training

Create the RouteRequest mediation flow in Exercise 11

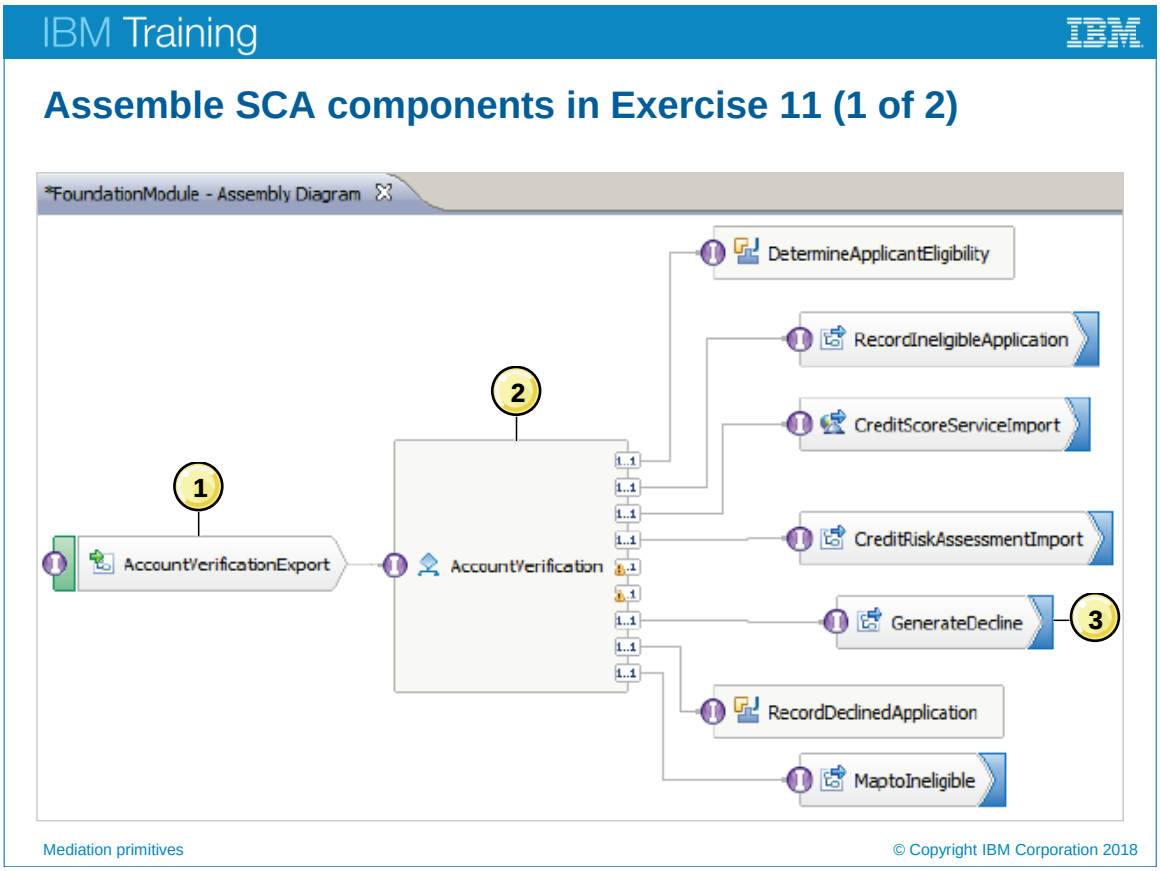


Mediation primitives

Create the RouteRequest mediation flow in Exercise 11

If applicationDecision is set to false during FinalApplicationReview (the application is declined) and the customer's creditRisk is HIGH, the application is routed through the "generate decline" component. If applicationDecision is set to false during FinalApplicationReview and the customer's creditRisk is MED (short for medium), the application is routed through the "special decline" component.

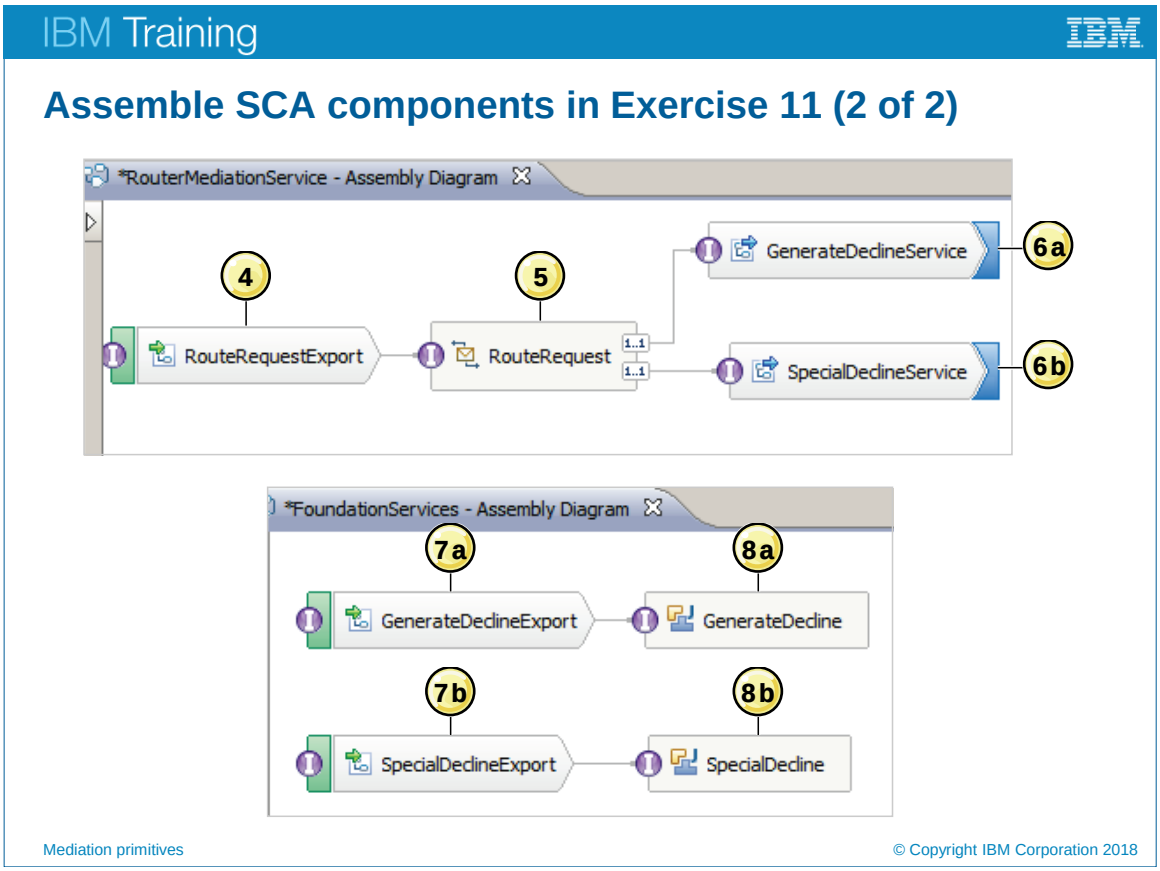
In this portion of the exercise, you implement the mediation flow for the RouteRequest component. The RouteRequest flow component contains the mediation logic that routes the application to the appropriate decline service. The RouteRequest mediation flow consists of both a request flow and a response flow. In the flow, the CustomerApplication is routed to the appropriate decline service by a router mediation primitive. After processing, the response from the decline service is sent back to the AccountVerification process.



Assemble SCA components in Exercise 11

The following steps are illustrated in the diagram:

- **Step 1**: The AccountVerificationExport component exposes the AccountVerification business process.
- **Step 2**: If the Application is declined link is followed, the Generate Decline invoke activity is processed. The Generate Decline activity uses the GenerateDeclinePartner reference partner to call the GenerateDecline import component on the FoundationModule assembly diagram.
- **Step 3**: The GenerateDecline import component invokes the RouteRequestExport component on the RouterMediationService assembly diagram.



The following steps are illustrated in the diagram:

- **Step 4**: RouteRequestExport exposes the services of the RouteRequest mediation flow.
- **Step 5**: If the creditRisk is HIGH and applicationDecision is false, the message is routed to the GenerateDeclineService import component. If the creditRisk is MED and applicationDecision is false, the message is routed to the SpecialDeclineService import component.
- **Step 6a**: The GenerateDeclineService import invokes the GenerateDeclineExport component on the FoundationServices assembly diagram.
- **Step 6b**: The SpecialDeclineService import invokes the SpecialDeclineExport component on the FoundationServices assembly diagram.
- **Step 7a**: The GenerateDeclineExport exposes the services of the GenerateDecline Java component.
- **Step 7b**: SpecialDeclineExport exposes the services of the SpecialDecline Java component.
- **Step 8a**: This component sets the message element of the Message business object to: Account for customer TestCo was declined and the credit risk was HIGH.

Exercise 11: Creating mediation services, part II

Purpose:
In this exercise, you use a message filter primitive to implement a mediation module that routes messages.

Mediation primitives are the building blocks of mediation flows in business integration modules and mediation modules. Mediation flows operate on messages that are in-flight between service requesters and service providers. You can use each mediation primitive to do different things with a message. Mediation primitives process messages as service message objects (SMOs) because SMOs allow different types of messages to be processed in a common way.

For example, different messages can use the message filter mediation primitive to take different paths. A message might need forwarding to different service providers based on the request details. You can also use the message content as the basis for bypassing unnecessary steps. If the criterion is not met, you can use the fail mediation primitive to raise a fault, or you can send an error response.

Requirements

Completing the exercises for this course requires a lab environment. This environment includes the exercise support files, IBM Process Designer, IBM Process Center, and IBM Integration Designer test environment.

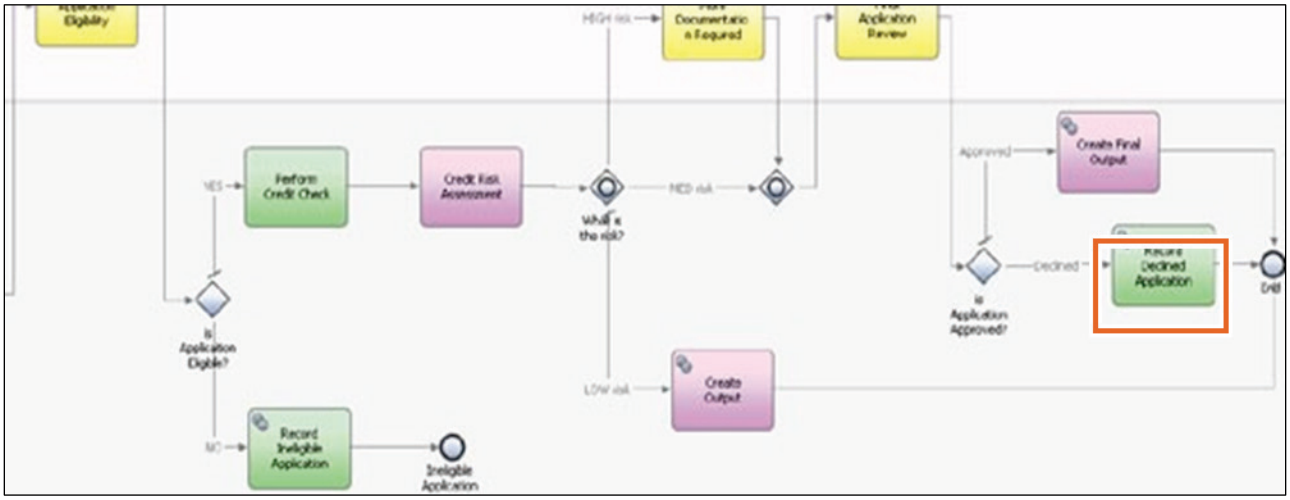
Exercise instructions

In this exercise, you implement a simple mediation module that is named `RouterMediationService` that is designed to route messages between services. A message filter primitive is used to route the messages. When you implement the mediation service, you wire it to the AccountVerification process component on the FoundationModule assembly diagram.

Remember in the account verification process narrative, a customer application can be declined. The approval and decline human task user interface is created in another exercise. If the customer's creditRisk is `MED` (short for medium), the application is routed directly to the FinalApplicationReview activity. If the customer's creditRisk is `HIGH`, more documentation is requested before the application is routed to the FinalApplicationReview activity.

If the employee approves the application, the process completes successfully. However, if the application is declined, it is routed to one of two possible services. If the application is declined and the creditRisk is `HIGH`, the generate decline service is invoked. If the application is declined and the creditRisk is `MED`, the special decline service is invoked. The implementation of both services uses simple Java code to return messages to the console that indicates which “declined” service was called. These services would normally be implemented as adapters or other types of services that would archive the applications for auditing or send them for more review.

It is captured in the process diagram:



Part 1. Create a mediation module that contains a message filter mediation primitive and a Mapping primitive.

In this portion of the exercise, you generate the implementation for the “generate decline” service. If the application is declined during FinalApplicationReview and the customer’s creditRisk is `HIGH`, the “generate decline” service is called.

To implement the “generate decline” service:

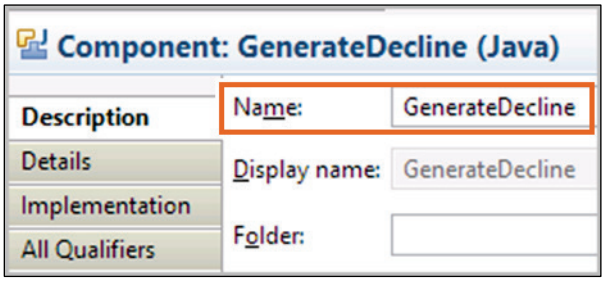
1. Open the Exercise 11 workspace.

1. On your desktop, open the **Exercise Shortcuts** folder.
2. Double-click the **Exercise 11** shortcut.

   Allow Integration Designer a few moments to build the workspace. You can view the workspace build status at the lower right corner of Integration Designer. Wait until the status reaches 100%, at which point the workspace is built, and the status progress bar disappears.
3. If a message that the server is already set to publish is displayed, click **OK**.
4. Close the **Getting Started** tab.

2. Create a Java component that is named GenerateDecline on the FoundationServices assembly diagram.

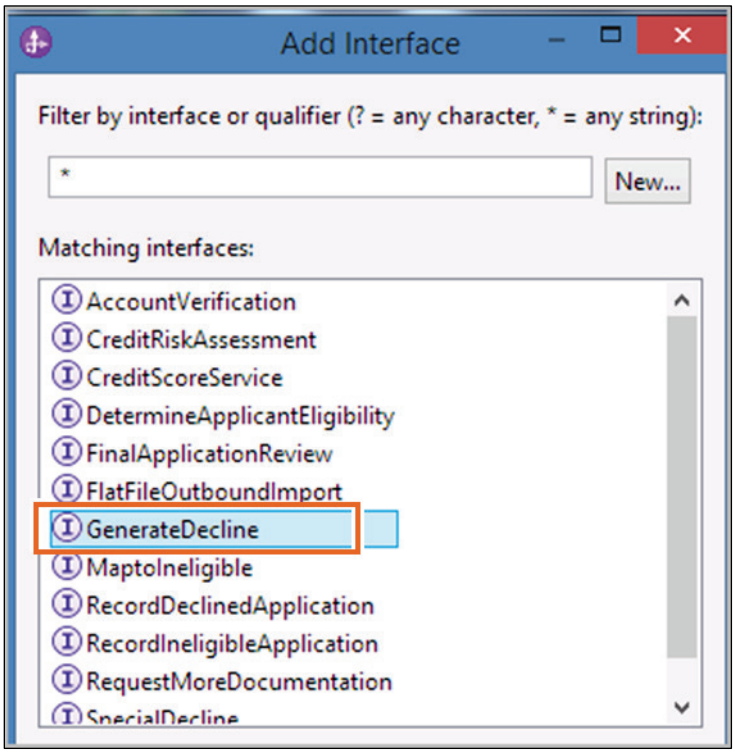
1. In the Business Integration view, expand **FoundationServices** and double-click **Assembly Diagram**.
2. In the palette, expand **Components** and select **Java**.
3. Click any blank space on the assembly diagram to add the Java component.
4. Switch to the **Description** tab in the **Properties** view.
5. Change the **Name** of the Java component to: `GenerateDecline`



6. Save your changes.

3. Add the GenerateDecline interface to the component.

1. Right-click the **GenerateDecline** Java component and click **Add > Interface** from the menu.
2. In the Add Interface dialog box, select **GenerateDecline**.

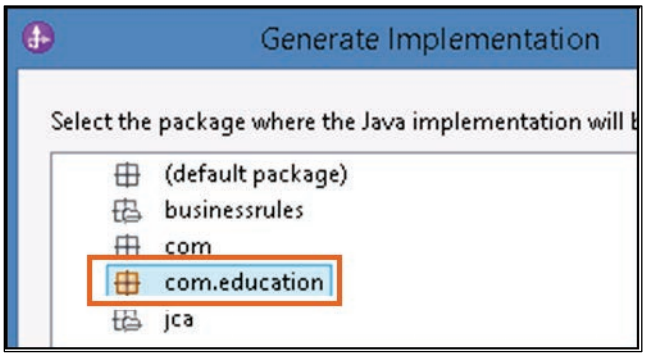


3. Click **OK**.

4. Use the code in C:\labfiles\Support Files\Ex11\GenerateDecline_InputCriterion.txt to create the implementation for the Java component.

The code returns a Message business object that contains the message: “Account for customer <company name> was declined and the credit risk was <credit risk>.”

1. Right-click GenerateDecline and click Generate Implementation from the menu.
2. In the Generate Implementation dialog box, select **com.education**.



3. Click **OK** to open `GenerateDeclineImpl.java` in the Java editor.
4. Open Windows Explorer and browse to `C:\labfiles\Support Files\Ex11`.
5. Open `GenerateDecline_InputCriterion.txt` in a text editor such as Notepad.
6. Copy the text that is in the file.
7. On the **GenerateDeclineImpl.java** tab, scroll to the `public DataObject InputCriterion` method at the end of the file.
8. Replace the comment lines (the green text that begins with `//`) with the code from `GenerateDecline_InputCriterion.txt`. Be sure to remove the `return null;` line from the method.
9. Alternatively, enter the following code:

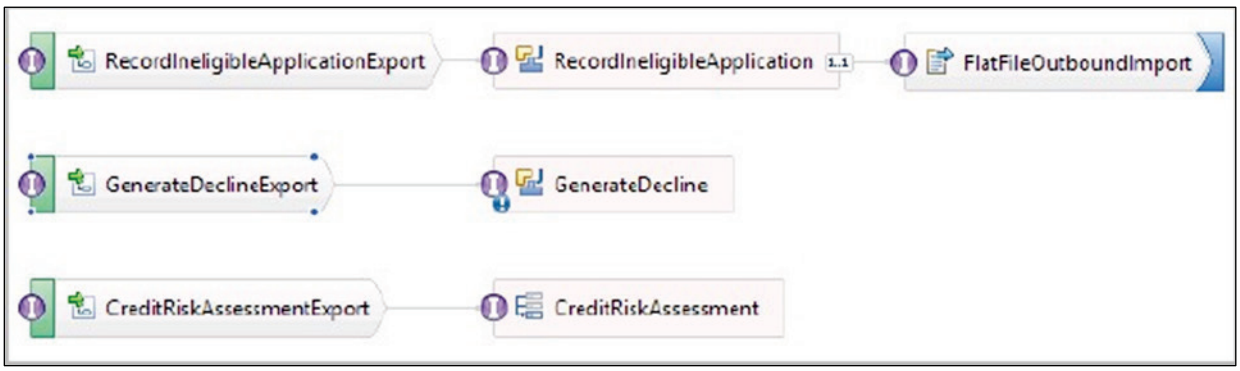
```
System.out.println("[Java] Generate Decline - begins");
String ret = "Account for customer " +
input.getString("companyName") + " was declined and the credit risk
was " + input.getString("creditRisk"); System.out.println("[Java]
Generate Decline - " + ret);
DataObject response =
com.ibm.websphere.sca.sdo.DataFactory.INSTANCE.create("http://Founda
tionL ibrary/businessitems", "Message");
response.setString("message", ret); System.out.println("[Java]
Generate Decline - ends"); return response;
```

10. Press Ctrl+S to save your changes. The left margin of the Java editor contains no error markers.
11. Close the Java editor.
12. Close the `GenerateDecline_InputCriterion.txt` file, but leave Windows Explorer open.

5. Generate an export component for GenerateDecline on the FoundationServices assembly diagram that is named GenerateDeclineExport. The export has an SCA binding.

1. On the **FoundationServices** assembly diagram, right-click **GenerateDecline** and click **Generate Export > SCA Binding** from the menu.
2. Accept the default export name: `GenerateDeclineExport`
3. Save your changes.

   Your assembly diagram resembles the following figure:



Implementing the special decline service

In this portion of the exercise, you generate the implementation for the “special decline” service. The “special decline” service is called when the application is declined during FinalApplicationReview and the creditRisk is `MED`.

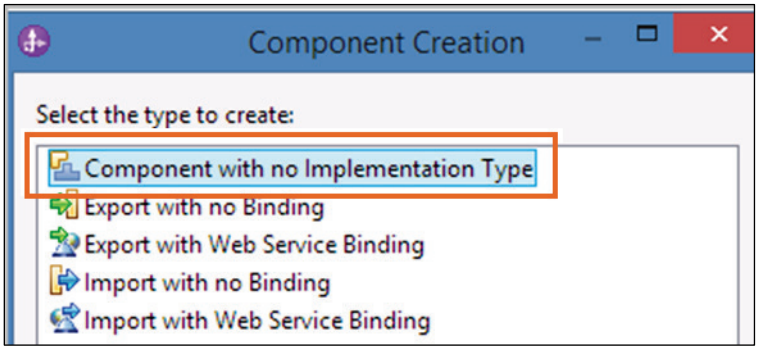
To implement the “special decline” service:

6. On the FoundationServices assembly diagram, create an untyped component that is named SpecialDecline that uses the SpecialDecline interface.

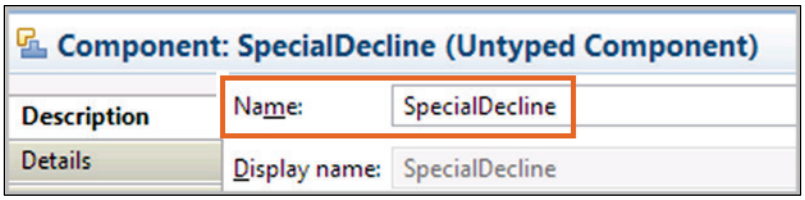
An untyped component can be used as a placeholder for future development. In his case, you generate the Java implementation immediately after creating the component. This exercise is done for educational purposes. You can also use a Java component instead of an untyped component.

1. In the Business Integration view, expand **FoundationLibrary > Interfaces**.

2. Drag the **SpecialDecline** interface onto the **FoundationServices** assembly diagram.
3. In the Component Creation dialog box, select **Component with no Implementation Type**.



4. Click **OK**.
5. Switch to the **Description** tab in the **Properties** view.
6. Change the **Name** of the component to: `SpecialDecline`

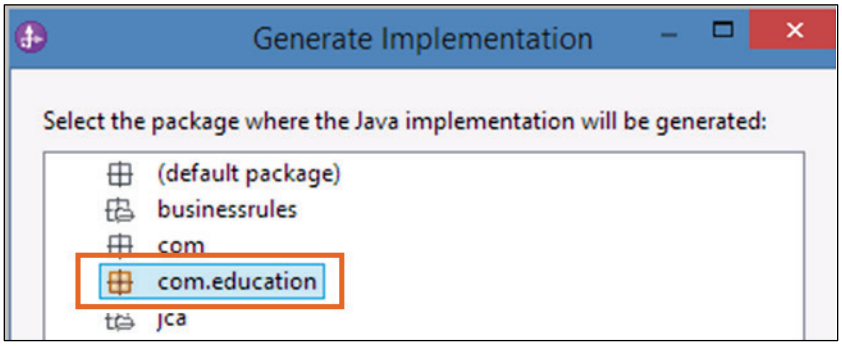


7. Save your changes.

7. Use the code in C:\labfiles\Support Files\Ex11\SpecialDecline_InputCriterion.txt to generate the Java implementation for the SpecialDecline component.

The code in SpecialDecline returns a Message business object that contains the message: “Account for customer <company name> was routed through special decline because the credit risk was <credit risk>.”

1. On the `FoundationServices` assembly diagram, right-click **SpecialDecline** and click **Generate Implementation > Java** from the menu.
2. In the **Generate Implementation** dialog box, select the **com.education** package.



3. Click **OK**.

   The `SpecialDeclineImpl.java` file opens in the Java editor.

4. In Windows Explorer, open `C:\labfiles\Support Files\Ex11\SpecialDecline_InputCriterion.txt` in a text editor such as Notepad.

5. Copy the text from `SpecialDecline_InputCriterion.txt`.

6. On the **SpecialDeclineImpl.java** tab, scroll to the `public DataObject InputCriterion` method at the end of the file.

7. Paste the code from `SpecialDecline_InputCriterion.txt` over the green comment lines (that begin with `//`). Be sure to remove `return null;` from the method.

   Alternatively, enter the following code:

```
public DataObject InputCriterion(DataObject input) {
System.out.println("[Java] Generate Decline Special - begins");
String ret = "Account for customer "
+ input.getString("companyName") + " was routed through special
decline because the credit risk was " +
input.getString("creditRisk"); System.out.println("[Java] Generate
Decline Special - " + ret); DataObject response =
com.ibm.websphere.sca.sdo.DataFactory.INSTANCE.create(
"http://FoundationLibrary/businessitems", "Message");
response.setString("message", ret);
System.out.println("[Java] Generate Decline Special - ends"); return
response;
```

8. Click **File > Save All** from the menu options.

9. Close the Java editor.

10. Close `SpecialDecline_InputCriterion.txt` and close Windows Explorer.

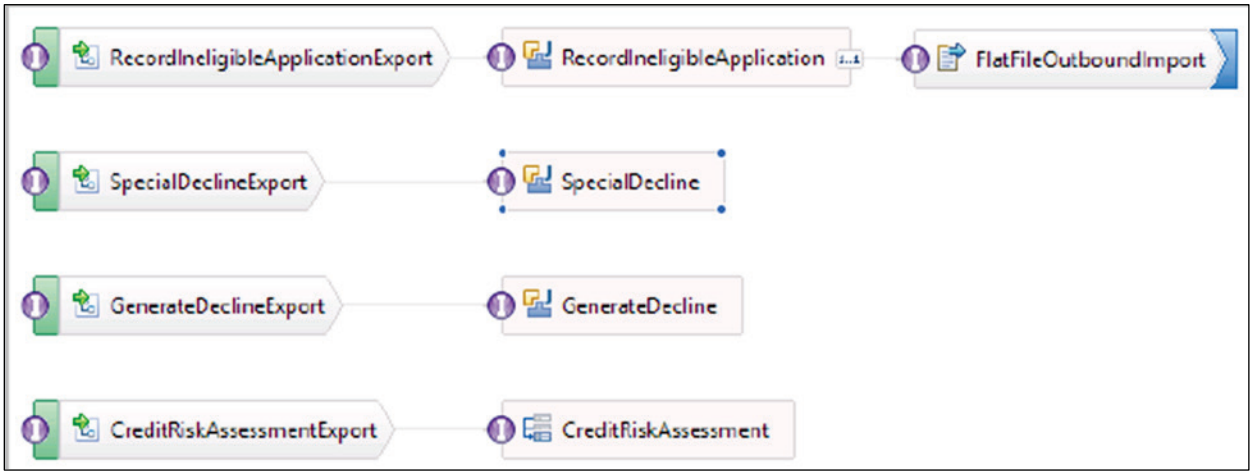
    You will generate an export component for SpecialDecline on the FoundationServices assembly diagram that is named `SpecialDeclineExport`. The export has an SCA binding.

11. On the **FoundationServices** assembly diagram, right-click **SpecialDecline** and click **Generate Export > SCA Binding** from the menu.

12. Accept the default export name: `SpecialDeclineExport`

13. Save your changes.

    Your assembly diagram resembles the following figure:



Implementing the RouteRequest mediation flow component

If applicationDecision is set to `false` during FinalApplicationReview (the application is declined) and the customer's creditRisk is `HIGH`, the application is routed through the "generate decline" component. If applicationDecision is set to `false` during FinalApplicationReview and the customer's creditRisk is `MED` (short for medium), the application is routed through the "special decline" component.

In this portion of the exercise, you implement the mediation flow for the RouteRequest component. The RouteRequest flow component contains the mediation logic that routes the application to the appropriate decline service.

The RouteRequest mediation flow consists of both a request flow and a response flow. In the flow, the CustomerApplication is routed to the appropriate decline service by a router mediation primitive. After processing, the response from the decline service is sent back to the AccountVerification process.

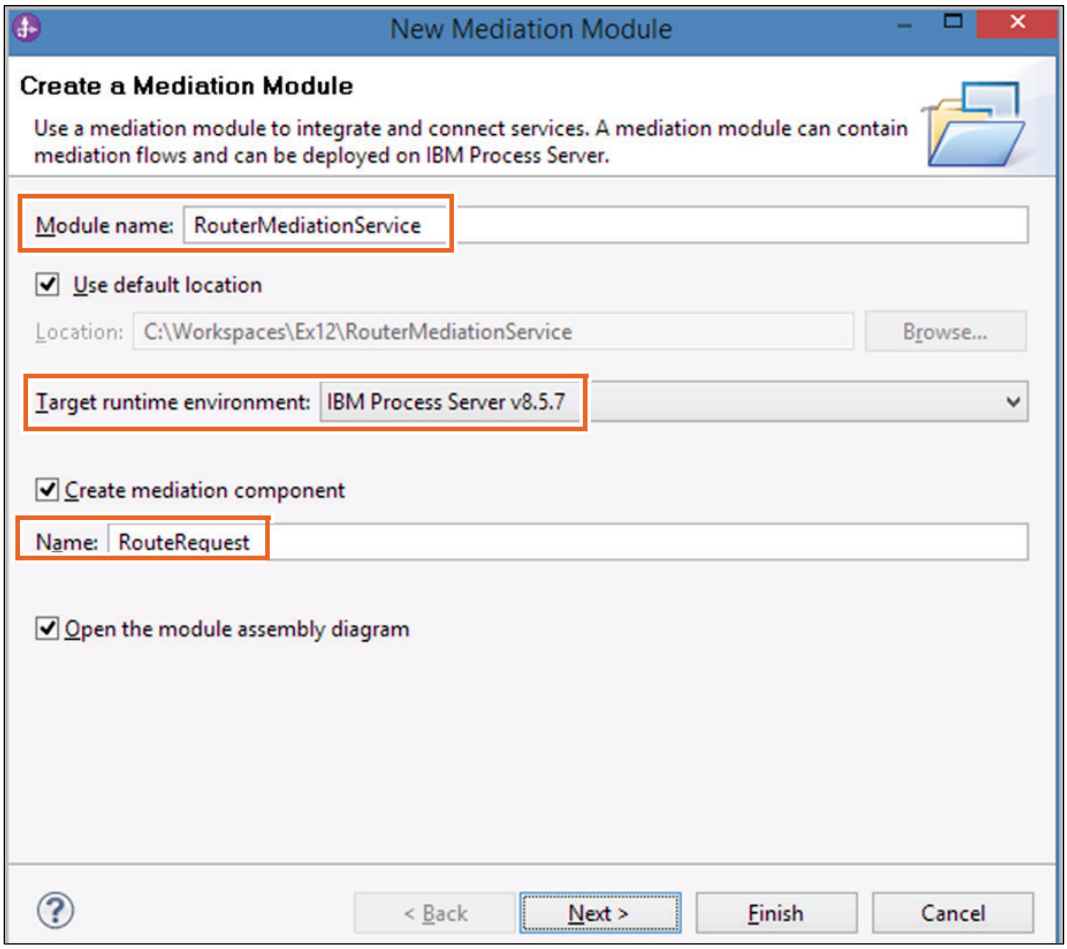
To implement the RouteRequest mediation flow component:

8. Create a mediation module that is named `RouterMediationService`.

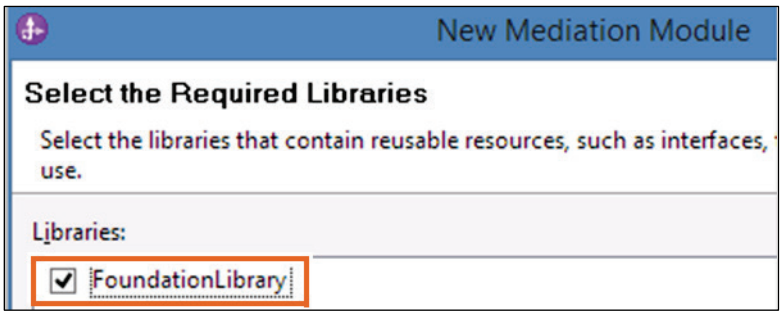
The target runtime environment for the module is **IBM Process Server v8.6** and the module has a dependency on **FoundationLibrary**. The mediation component in the module is named `RouteRequest`.

1. Click File > New > Mediation Module from the menu options.

2. In the “Create a Mediation Module” window, enter the following information.
   - In the **Module name** field, type: `RouterMediationService`
   - In the **Target runtime environment** field, select **IBM Process Server v8.6**.
   - Change the **Name** of the mediation component to: `RouteRequest`

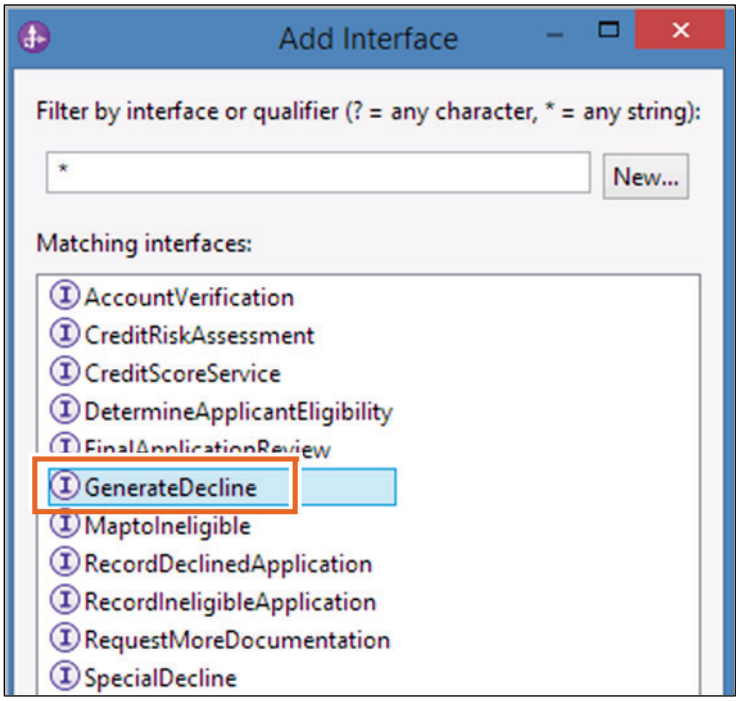


3. Accept the remaining default options and click **Next**.
4. In the “Select required libraries” window, select the **FoundationLibrary** check box to include it as a dependency.



5. Click **Finish**.

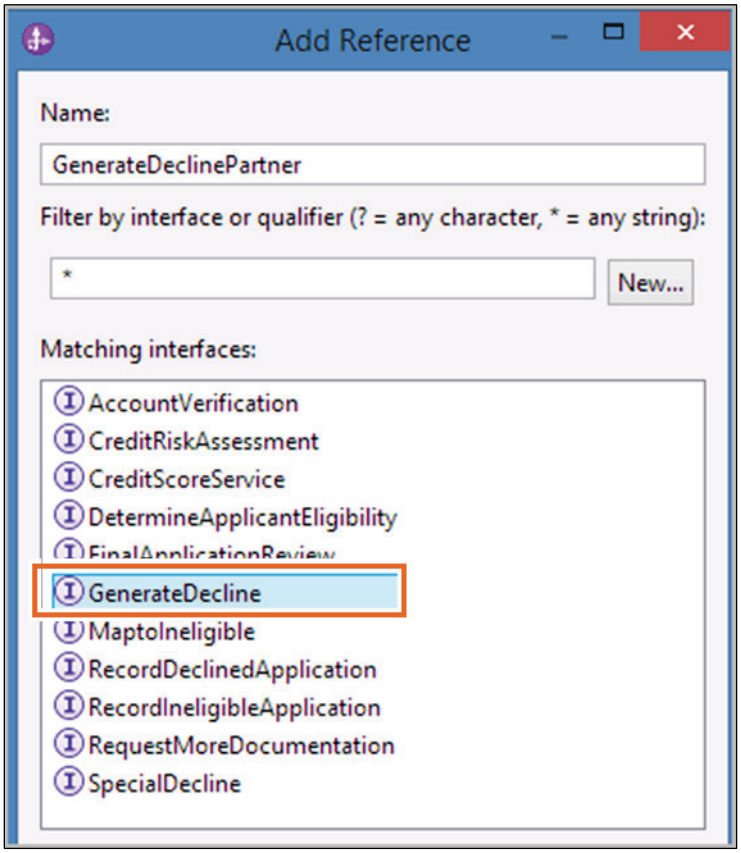
9. Add the GenerateDecline interface to the RouteRequest component.
   1. On the **RouterMediationService** assembly diagram, right-click the **RouteRequest** component and click **Add > Interface** from the menu.
   2. In the Add Interface dialog box, select the **GenerateDecline** interface.



   3. Click **OK**.
   4. Save your changes.

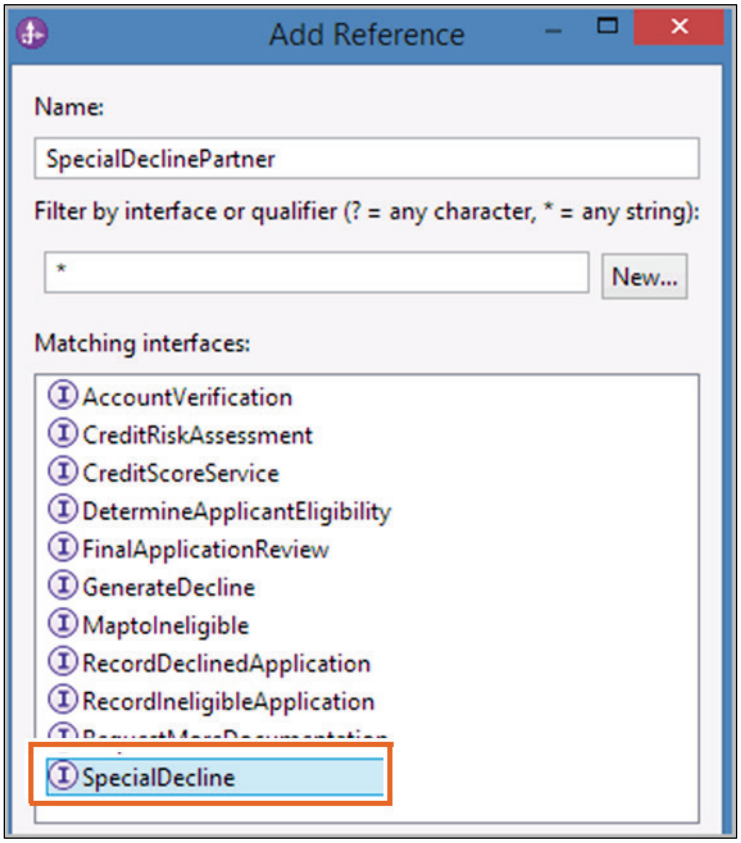
10. Add two references to the RouteRequest flow component. Add the GenerateDeclinePartner reference that uses the GenerateDecline interface and add the SpecialDeclinePartner that uses the SpecialDecline interface.
    1. On the **RouterMediationService** assembly diagram, right-click the **RouteRequest** component and click **Add > Reference**.

2. In the Add Reference dialog box, select the **GenerateDecline** interface.

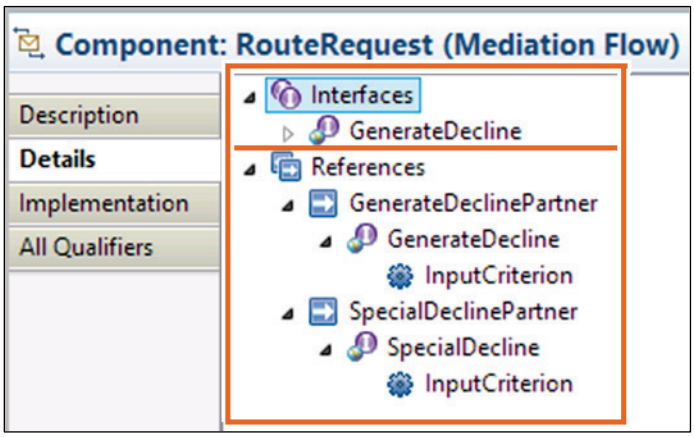


3. Click **OK**.
4. On the **RouterMediationService** assembly diagram, right-click the **RouteRequest** component and click **Add > Reference**.

5. In the Add Reference dialog box, select the **SpecialDecline** interface.



6. Click **OK**.
7. Switch to the **Details** tab in the **Properties** view.
8. Verify the interface and references for the **RouteRequest** flow component.

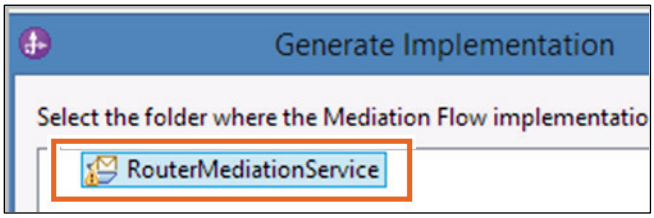


9. Save your changes.

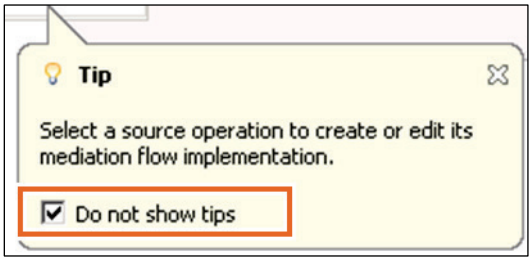
Part 2. Define an XML data map.

1. Use the Blank Mediation Flow template to generate the implementation for the RouteRequest request flow component in the RouterMediationService folder.

1. Right-click the **RouteRequest** flow component and click **Generate Implementation**.
2. In the Generate Implementation dialog box, leave the **RouterMediationService** folder selected.

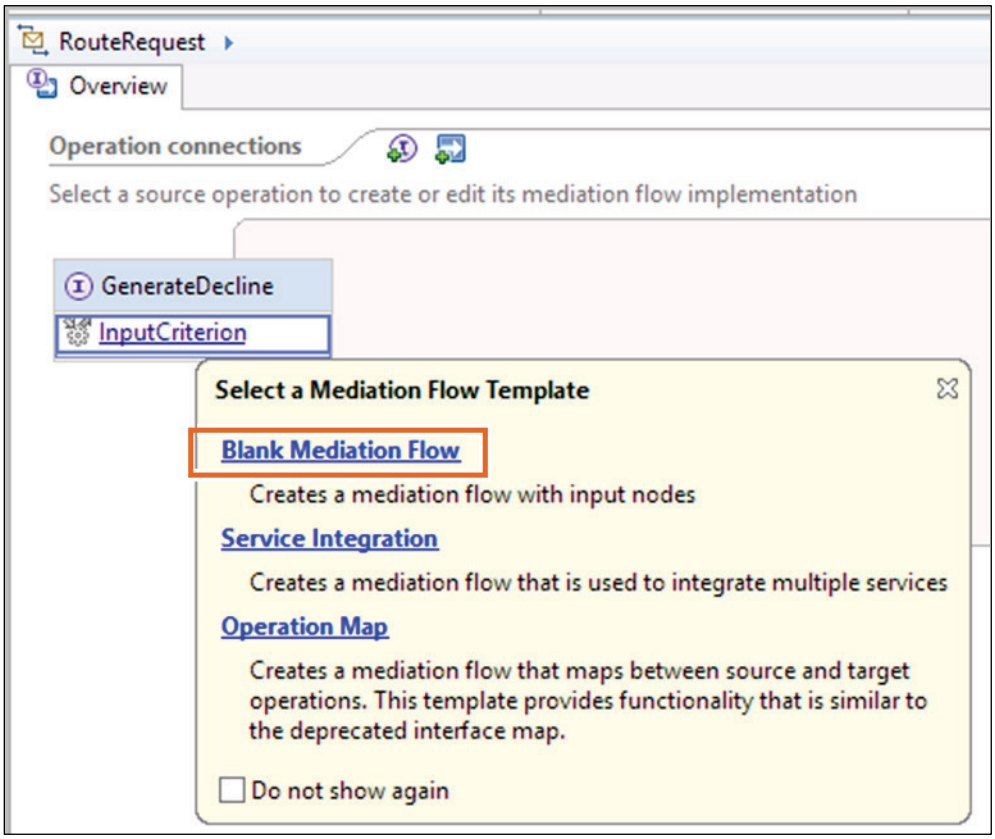


3. Click **OK**.
4. In the mediation flow editor, select the **Do not show tips** check box in the Tip dialog box.



5. Close the Tip dialog box.

6. In the Operation connections section of the editor, click the **InputCriterion** operation in the **GenerateDecline** interface.



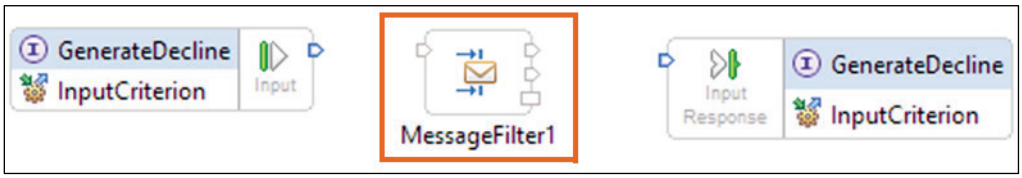
7. In the “Select a Mediation Flow Template” dialog box, click the **Blank Mediation Flow** link.

   The generated request flow consists of an **Input** node and an **Input Response** node. The **Input Response** node might be off screen to the right.

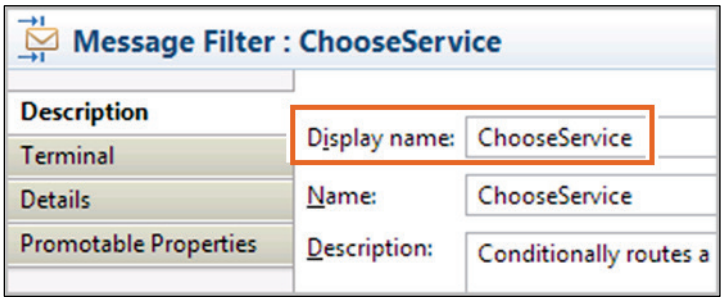
   You will add a **Message Filter** primitive in the request flow named `ChooseService`.

8. In the palette, expand the **Routing** folder and click **Message Filter**.

9. Click the blank space on the canvas between the **Input** and **Input Response** nodes to add the **Message Filter** primitive to the flow.



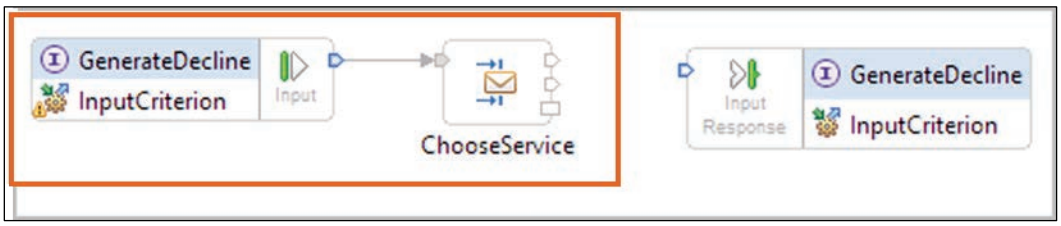
10. With the **Message Filter** primitive selected, switch to the **Description** tab in the **Properties** view.
11. Change the **Display Name** to: `ChooseService`



12. Save your changes.

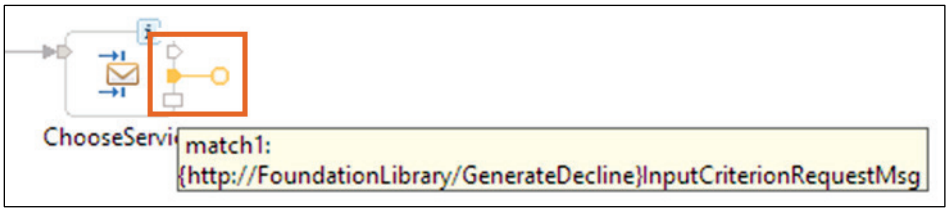
2. Wire the out terminal of the input node to the in terminal of the ChooseService primitive.

1. Right-click the **out** terminal of the **InputCriterion:GenerateDecline** input node and select **Add Connection** from the menu.
2. Click the **in** terminal of the **ChooseService** primitive to add the connection.

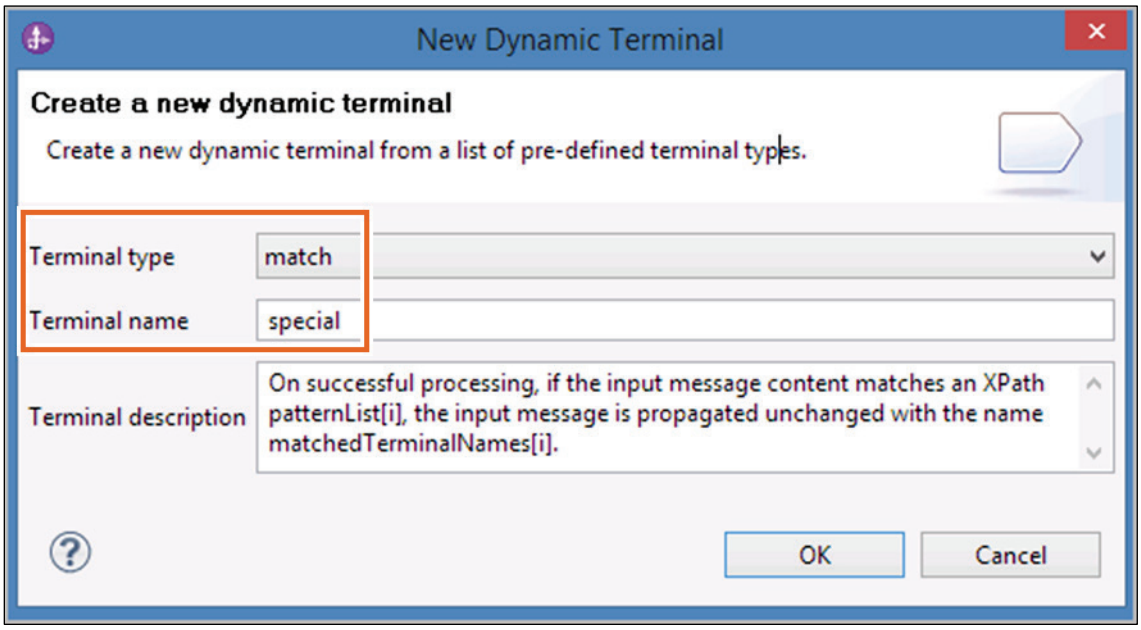


3. Delete the match1 terminal on the ChooseService primitive and create an out terminal that is named: `special`

1. Hover over each outbound terminal on **ChooseService** until you locate the **match1** terminal. The terminal name is listed in the descriptor.



2. Right-click the **match1** terminal and click **Delete** from the menu.
3. Right-click the **ChooseService** primitive and click **Add Output Terminal** from the menu.
4. In the New Dynamic Terminal dialog box, take the following actions.
   - Leave the **Terminal type** set to: `match`
   - Change the **Terminal name** to: `special`



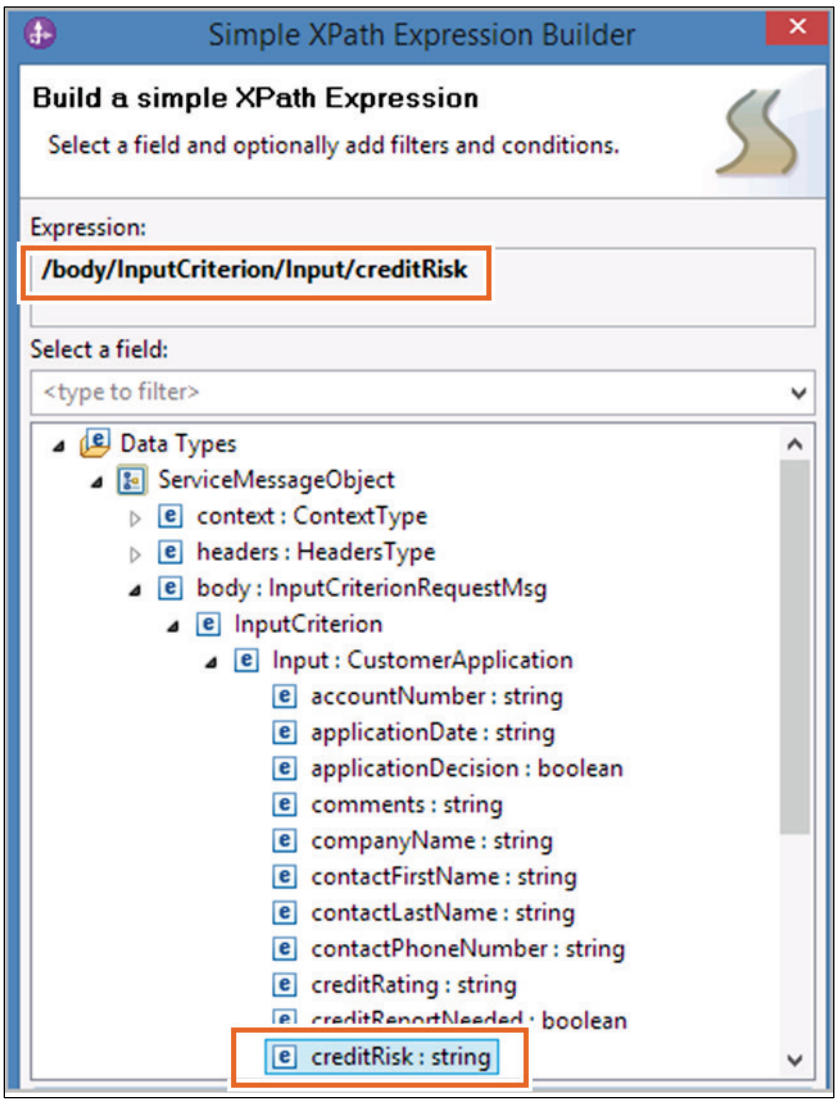
5. Click **OK**.

4. Define a pattern on the ChooseService primitive that passes control to the special out terminal when the incoming creditRisk is `MED`.

The XPath expression that the pattern uses is:
`/body/InputCriterion/Input/creditRisk='MED'`

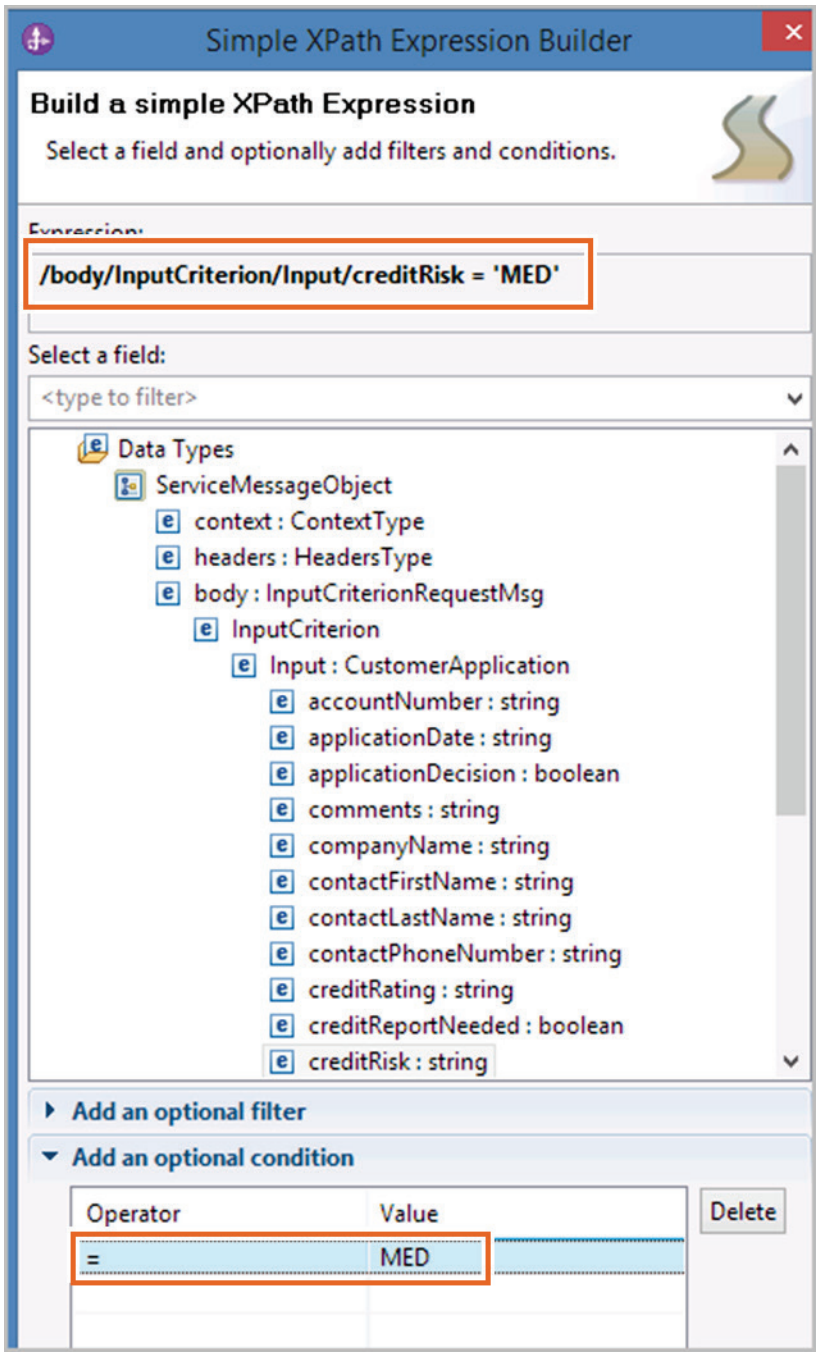
1. Select the **ChooseService** primitive and switch to the **Details** tab in the **Properties** view.
2. Click **Add** to add a pattern.
3. In the “Add/Edit properties” dialog box, by the **Pattern** field, click **Edit** to open the **XPath Expression Builder**.
4. In the “Build a simple XPath Expression” window, in the “Select a field” window, expand **Data Types > ServiceMessageObject > body: InputCriterionRequestMessage > InputCriterion > Input: CustomerApplication**.

5. Select **creditRisk: string** to see `/body/InputCriterion/Input/creditRisk` in the **Expression** field.



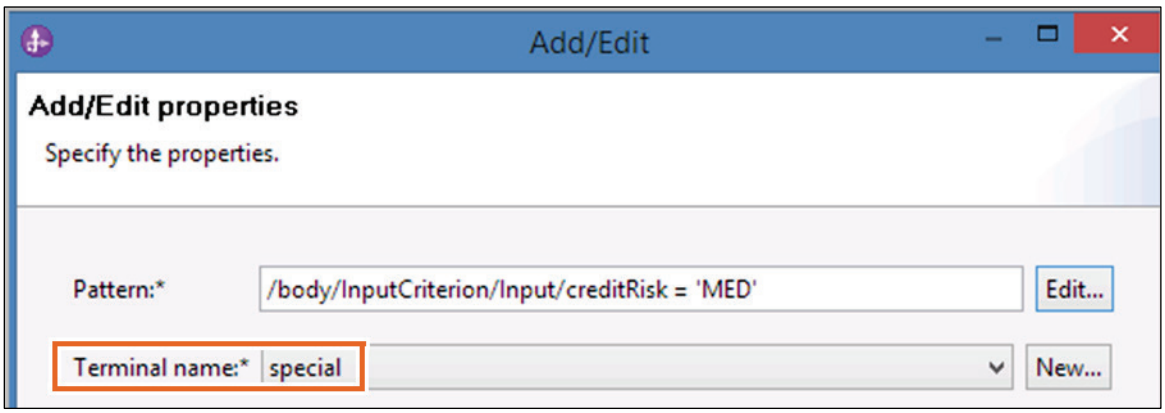
6. In the **Add an optional condition** section, leave **Operator** set to **=**, place the cursor in the **Value** column, type `MED`, and press Enter.

   The **Expression** field contains:
   `/body/InputCriterion/Input/creditRisk='MED'`

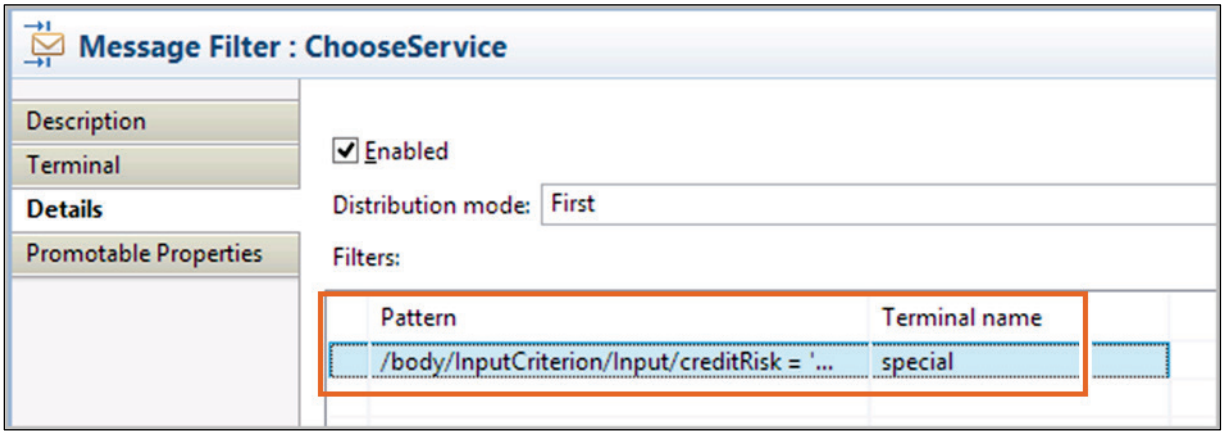


7. Click **OK**.

8. The **Pattern** field is populated with the XPath expression that you built for the **special** terminal. Verify that `special` is listed in the **Terminal name** field.



9. Click **Finish**.
   The expression is added to the **Pattern** list in the **Properties** view.



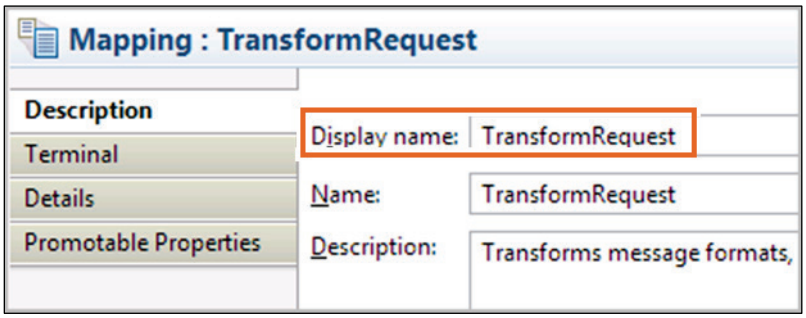
10. Save your changes.

5. Add a Mapping primitive that is named TransformRequest to the right of the ChooseService primitive.

Because of the disparity between the GenerateDecline and SpecialDecline interfaces, you must create a data map before the SpecialDeclinePartner callout node can be invoked.

1. In the palette, expand **Transformation** and click **Mapping**.
2. Click an empty area of the canvas to the right of the **ChooseService** primitive.
3. With the Mapping primitive selected, switch to the **Description** tab in the **Properties** view.

4. Change the **Display name** to: `TransformRequest`

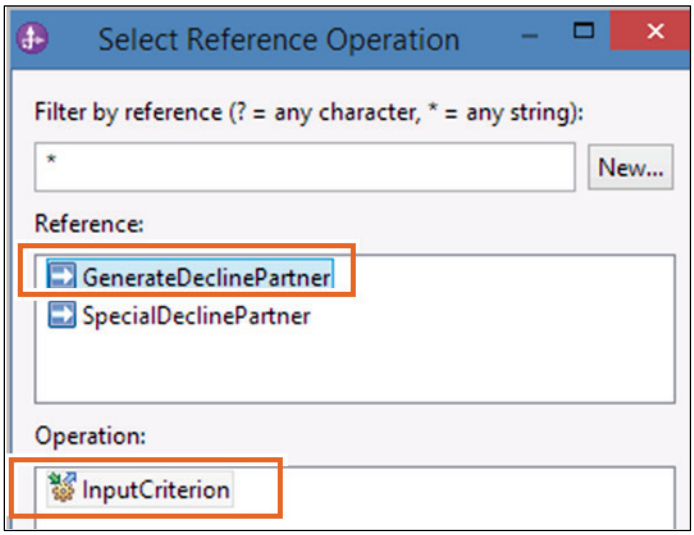


5. Save your changes.

6. Add two callout nodes to the request flow. One callout node is for the InputCriterion operation on the SpecialDecline interface of SpecialDeclinePartner.

   The second is for the InputCriterion operation on the GenerateDecline interface of GenerateDeclinePartner.

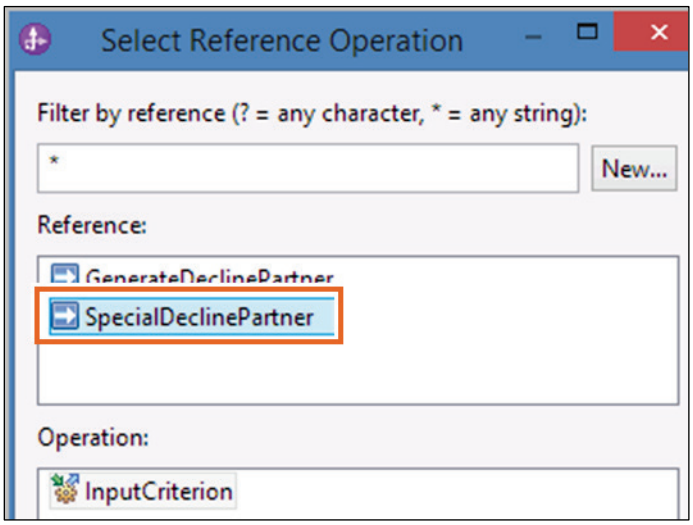
   1. In the palette, expand **Service Invocation** and click **Callout**.
   2. Click a blank area on the canvas to the right of the **ChooseService** primitive.
   3. In the Select Reference Operation dialog box, select the **GenerateDeclinePartner** reference.
   4. Because the interface has only one operation, **InputCriterion** is automatically selected.



   5. Click **OK**.
   6. In the palette, expand **Service Invocation** and click **Callout**.
   7. Click a blank area on the canvas to the right of the **ChooseService** primitive.

8. In the Select Reference Operation dialog box, select the **SpecialDeclinePartner** reference.

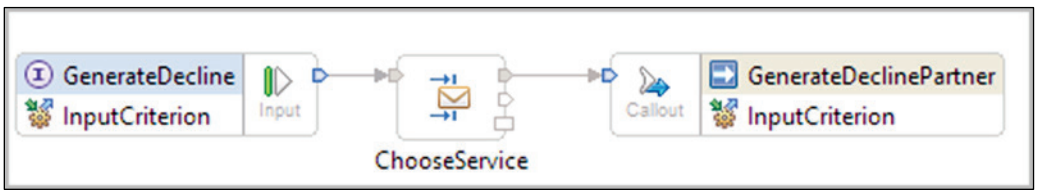
   Because the interface has only one operation, **InputCriterion** is automatically selected.



9. Click **OK**.
10. For readability, right-click the canvas and click **Layout Contents** from the menu.

7. Wire the default terminal of the ChooseService primitive to the in terminal of the GenerateDeclinePartner callout node.

1. Right-click the **default** terminal on the **ChooseService** primitive and click **Add Connection** from the menu.
2. Click the **in** terminal of the **GenerateDeclinePartner** callout node to add the connection.

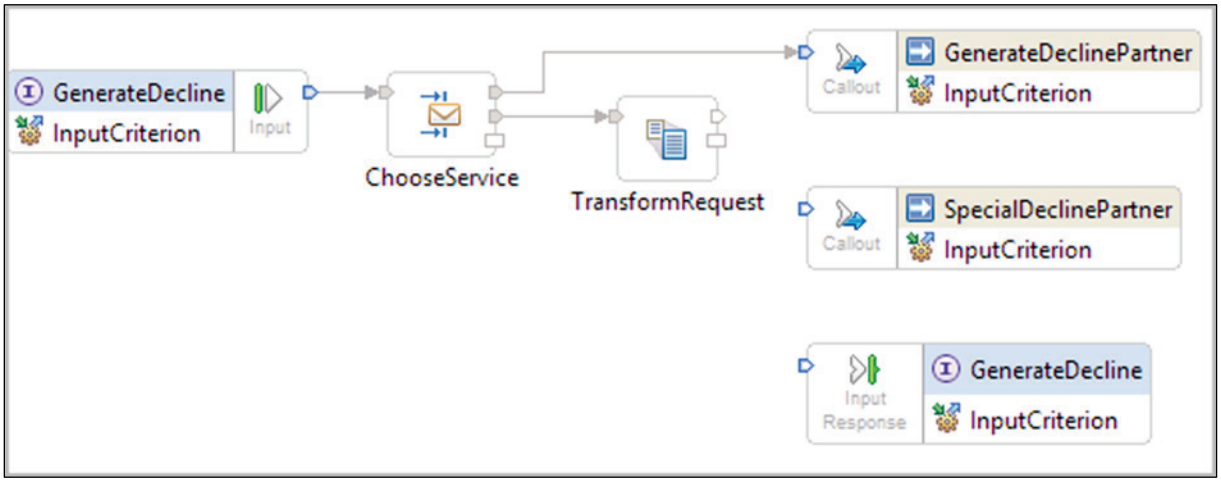


3. Save your changes.

8. Wire the special terminal of the ChooseService primitive to the in terminal of the TransformRequest primitive.

1. Right-click the **special** terminal on the **ChooseService** primitive and click **Add Connection** from the menu.

2. Click the **in** terminal of the **TransformRequest** primitive to add the wire.



3. Save your changes.

9. Wire the out terminal of the TransformRequest primitive to the in terminal of the SpecialDeclinePartner callout node.

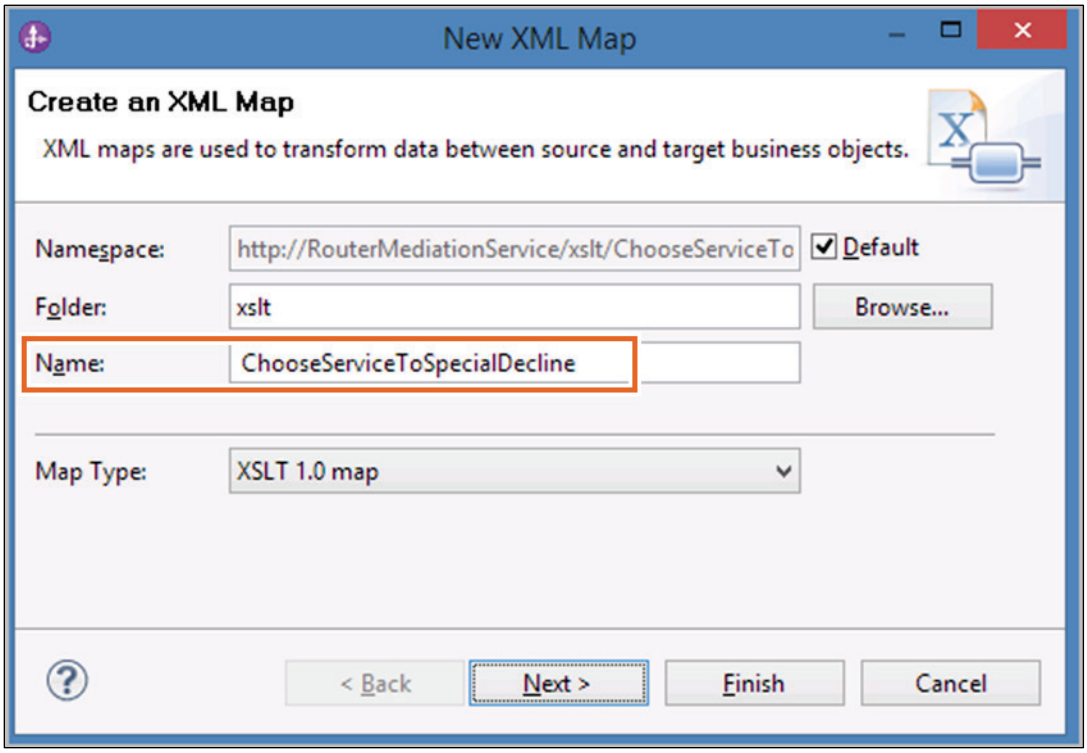
1. Right-click the **out** terminal on the **TransformRequest** primitive and click **Add Connection** from the menu.
2. Click the **in** terminal of the **SpecialDeclinePartner** callout node to add the connection.
3. Save your changes.

10. Implement the TransformRequest XML data map to transform the message from the ChooseService primitive to the SpecialDeclinePartner callout.

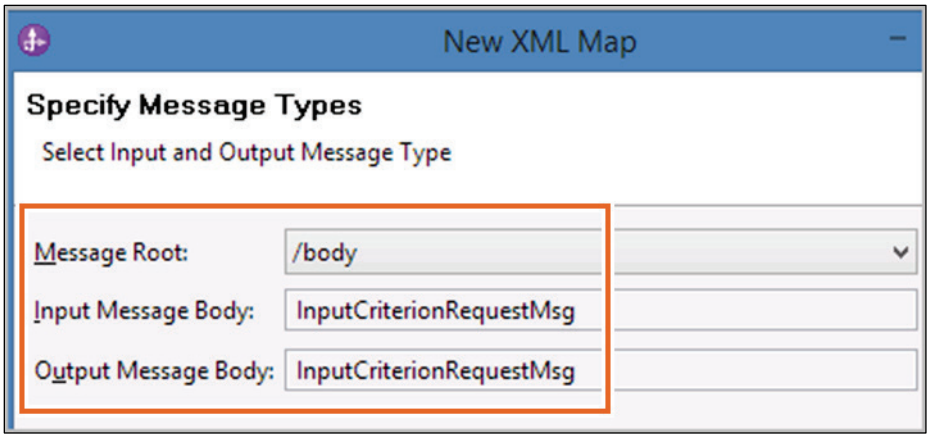
The map uses a local map to move the contents of the input SMO to the contents of the output SMO.

1. Select the **TransformRequest** primitive and switch to the **Details** tab in the **Properties** view.
2. By the **Mapping File** field, click **New** to create a map.

3. In the “Create an XML Map” window, take the following actions:
   - Verify that `xslt` is listed in the **Folder** field.
   - Change the **Name** to: `ChooseServiceToSpecialDecline`

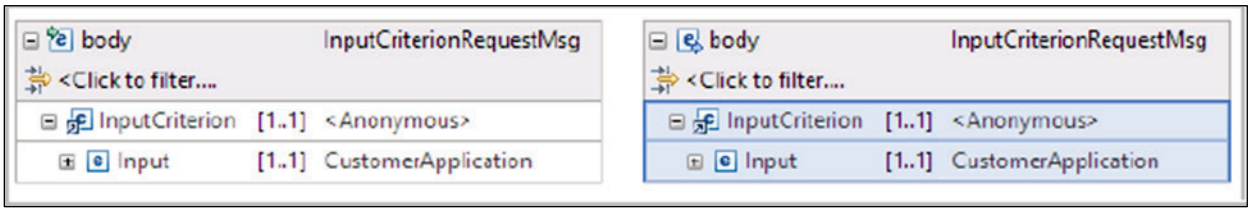


4. Accept the remaining default options and click **Next**.
5. In the Specify Message Types window, verify the following information:
   - **Message Root** is set to: `/body`
   - **Input Message Body** is set to: `InputCriterionRequestMsg`
   - **Output Message Body** is set to: `InputCriterionRequestMsg`

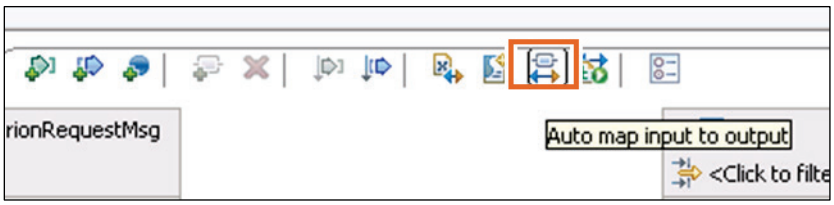


6. Ignore any warnings in the dialog box and click **Finish** to open the map editor.

7. In the map editor, expand **InputCriterion** in the input SMO and **InputCriterion** in the output SMO.



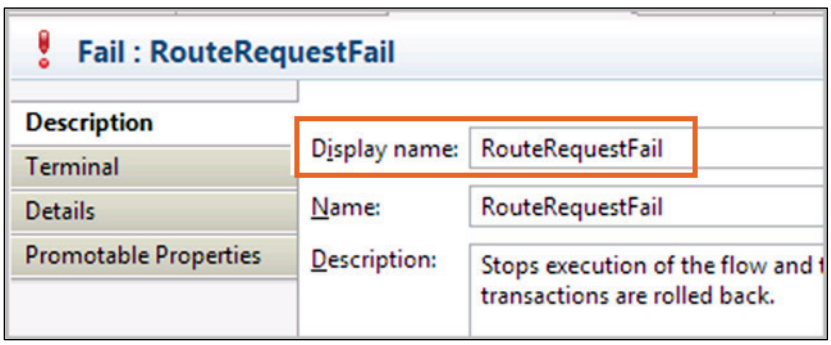
8. Click the **Auto map input to output** icon.



9. Accept the defaults in the Auto Map window and click **Finish**.
10. Click **File > Save All** from the menu options.
    The **Problems** view contains no errors. You can ignore any warnings.
11. Close the mapping editor.

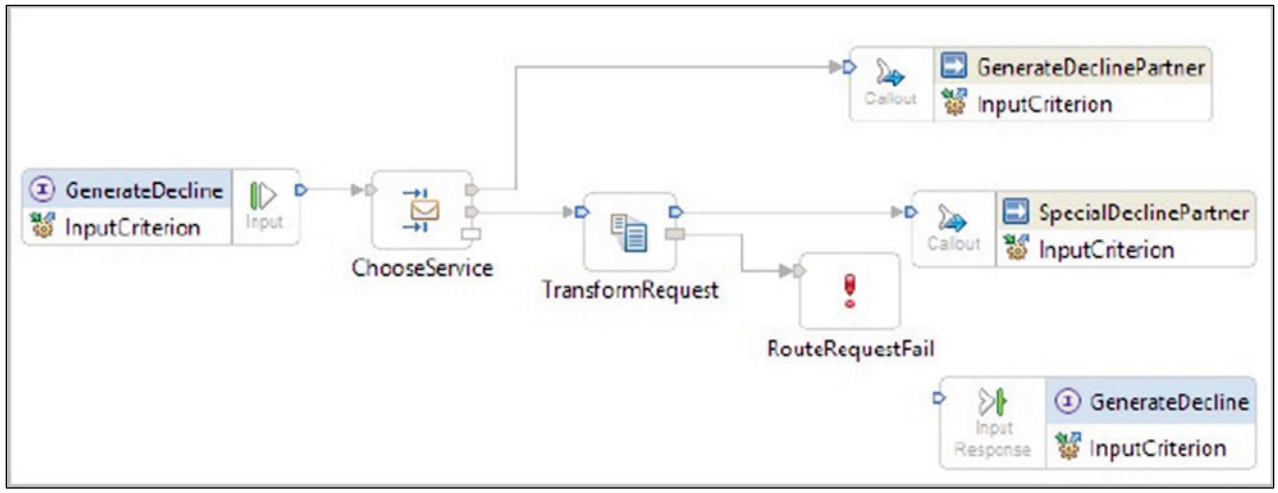
11. Add a Fail primitive that is named `RouteRequestFail` to the RouteRequest request mediation flow and wire it to the fail terminal on the TransformRequest primitive. If a failure is encountered in the map, the fail primitive stops flow execution and throws an exception.

1. In the palette, expand **Error Handling** and select **Fail**.
2. Click an empty portion of the flow diagram to the right of the **TransformRequest** primitive.
3. With the fail primitive selected, switch to the **Description** tab in the **Properties** view.
4. Change the **Display Name** to: `RouteRequestFail`

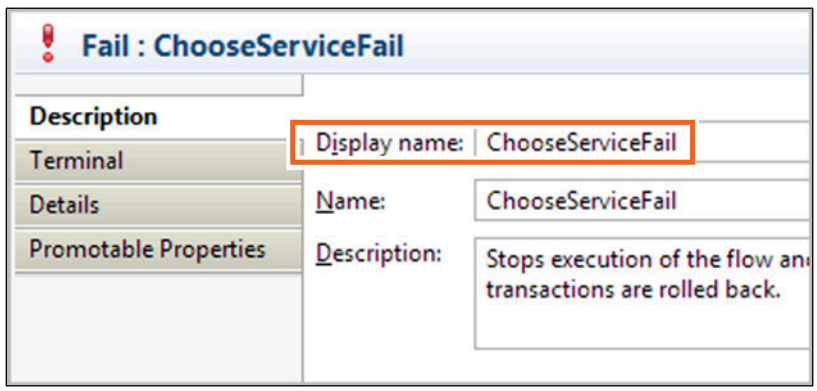


5. Right-click the **fail** terminal on the **TransformRequest** primitive and click **Add Connection** from the menu.
6. Click the **in** terminal on the **RouteRequestFail** primitive to add the connection.
7. To aid readability, right-click the flow diagram and click **Layout Contents** from the menu.

   The flow diagram resembles the following figure:

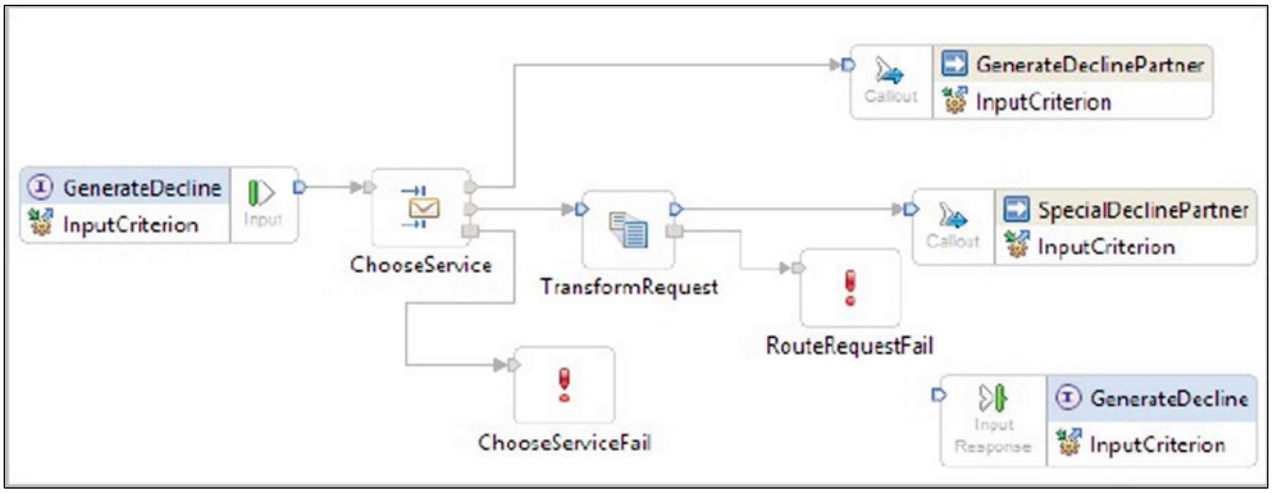


12. Add a Fail primitive named ChooseServiceFail to the RouteRequest request mediation flow and wire it to the fail terminal on the ChooseService primitive. If a failure is encountered in the message filter, the fail primitive stops flow execution and throws an exception.
    1. In the palette, expand **Error Handling** and select **Fail**.
    2. Click an empty portion of the flow diagram under the **ChooseService** primitive.
    3. With the fail primitive selected, switch to the **Description** tab in the **Properties** view.
    4. Change the **Display Name** to: ChooseServiceFail

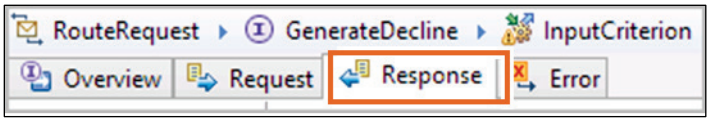


5. Right-click the **fail** terminal on the **ChooseService** primitive and click **Add Connection** from the menu.
6. Click the **in** terminal on the **ChooseServiceFail** primitive to add the connection.
7. To aid readability, right-click the flow diagram and click **Layout Contents** from the menu.

   The request flow diagram resembles the following figure:

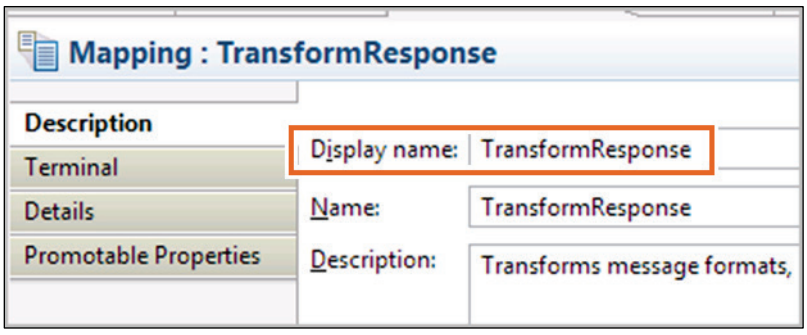


8. In the mediation flow editor, click the **Response** tab. You now implement the **response** message flow for the mediation.



13. Add a Mapping primitive named TransformResponse to the left of the Input Response node.

1. In the palette, expand **Transformation** and click **Mapping**.
2. Click the empty space on the canvas to the left of the **Input Response** node.
3. Switch to the **Description** tab in the **Properties** view.
4. Change the **Display Name** to: `TransformResponse`

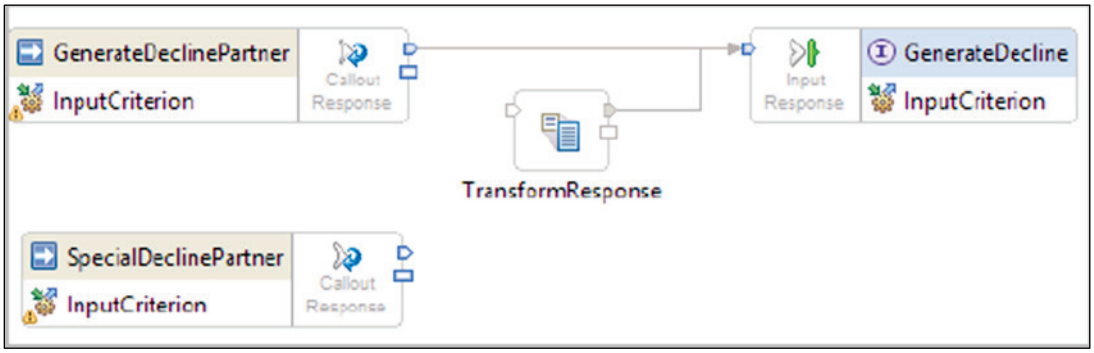


5. Save your changes.

14. Wire the out terminal of the TransformResponse primitive to the in terminal of the GenerateDecline Input Response node.
    1. Right-click the **out** terminal of the **TransformResponse** primitive and click **Add Connection** from the menu.
    2. Click the **in** terminal of the **GenerateDecline** Input Response node to add the connection.

15. Wire the out terminal of the GenerateDeclinePartner Callout Response node to the in terminal of the GenerateDecline Input Response node.
    1. Right-click the **out** terminal of the **GenerateDeclinePartner** Callout Response node and click **Add Connection** from the menu.
    2. Click the **in** terminal of the **GenerateDecline** Input Response node to add the connection.

    Your diagram resembles the following figure:

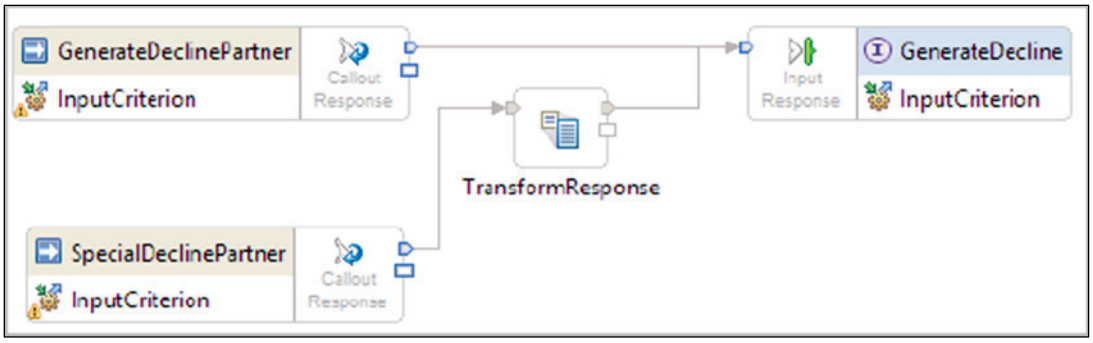


16. Wire the out terminal of the SpecialDeclinePartner Callout Response node to the in terminal of the TransformResponse primitive.
    1. Right-click the **out** terminal of the **SpecialDeclinePartner** Callout Response node and click **Add Connection** from the menu.

2. Click the **in** terminal of the **TransformResponse** primitive to add the connection.
3. Save your changes.

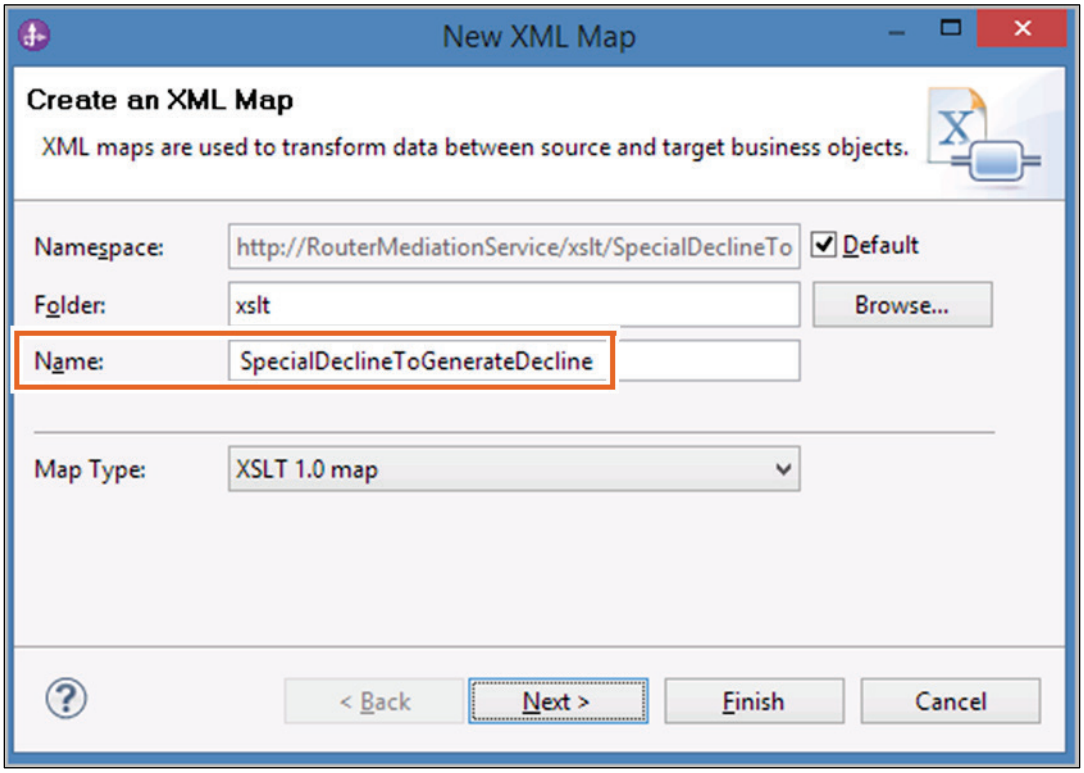
   Continue ignoring any errors in the **Problems** view.
4. For readability, right-click the flow diagram and click **Layout Contents** from the menu.

   Your diagram resembles the following figure:

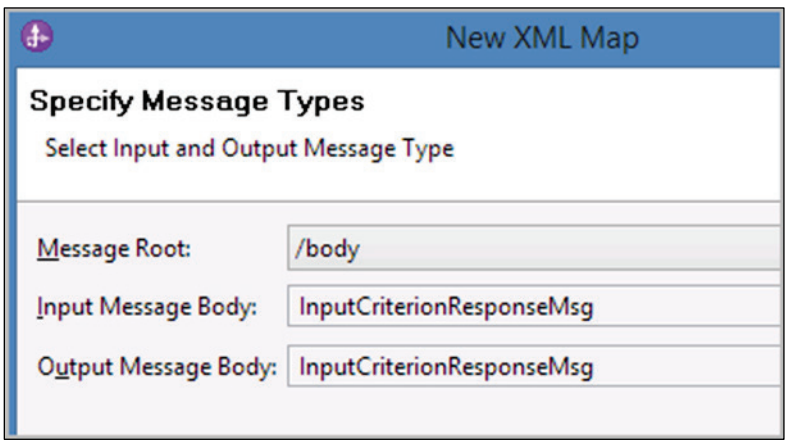


17. Implement the TransformResponse XML map to transform the input message from the SpecialDeclinePartner Callout Response to the GenerateDecline Input Response. The map uses a Move transformation to move the contents of the input SMO to the contents of the output SMO.

1. Select the **TransformResponse** primitive and switch to the **Details** tab in the **Properties** view.
2. By the **Mapping File** field, click **New** to create a map.
3. In the “Create an XML Map” window, do the following steps.
   - Verify that the **Folder** field is set to: `xslt`
   - Change the **Name** to: `SpecialDeclineToGenerateDecline`

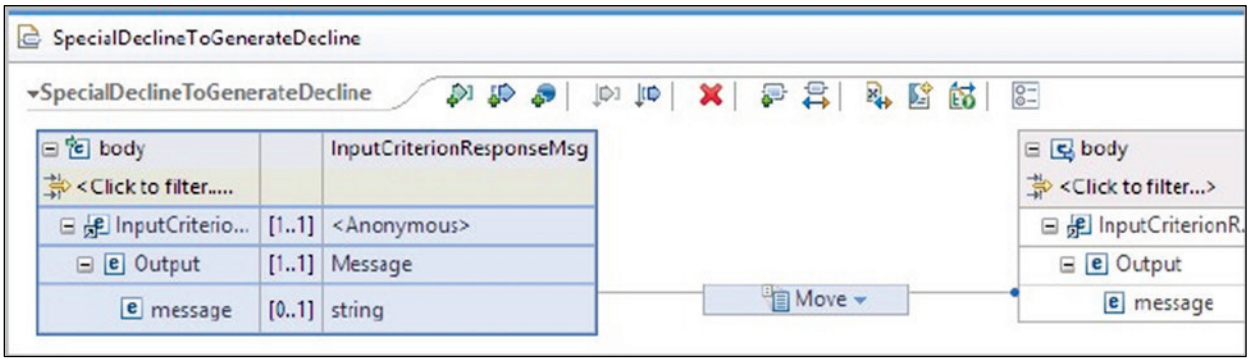


4. Click **Next**.
5. In the Specify Message Types window, verify the following information:
   - Message Root: `/body`
   - Input Message Body: `InputCriterionResponseMsg`
   - Output Message Body: `InputCriterionResponseMsg`



6. Ignore any warnings in the dialog box and click **Finish**.
7. Expand **InputCriterionResponse > Output** in the input SMO and expand **InputCriterionResponse > Output** in the output SMO.
8. In the input SMO, right-click the **message** attribute and click **Create Connection** from the menu.

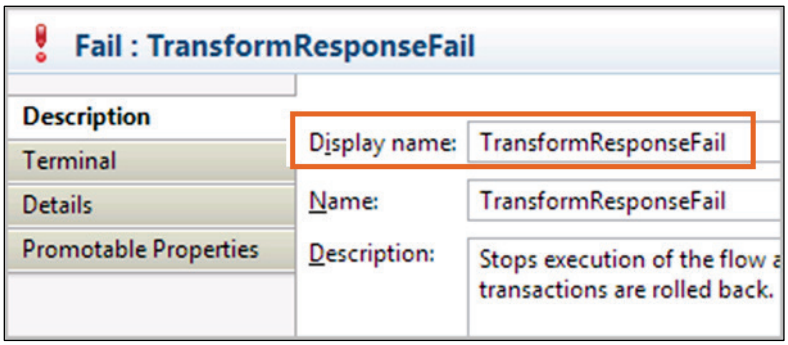
9. Click the **message** attribute in the output SMO to create the **Move** transformation.



10. Click **File > Save All** from the menu options.
11. Close the mapping editor.

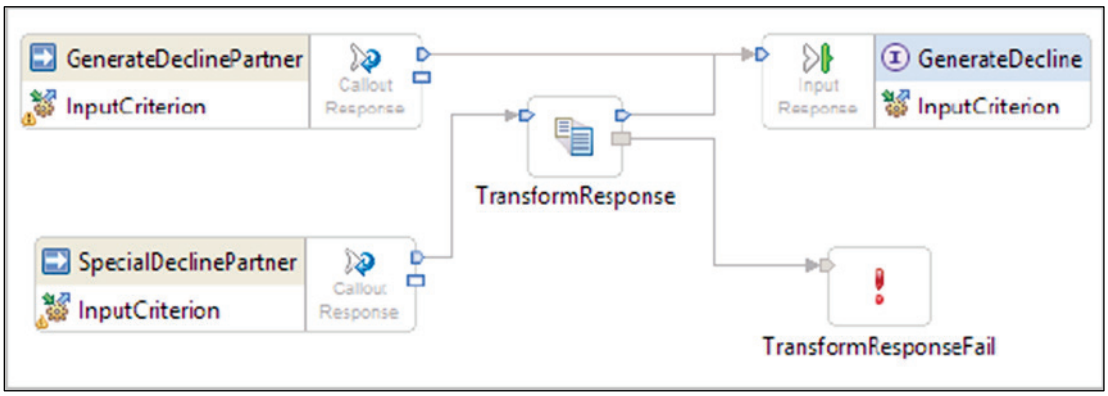
18. Add a Fail primitive that is named `TransformResponseFail` to the RouteRequest response mediation flow and wire it to the fail terminal on the TransformResponse primitive. If a failure is encountered in the map, the fail primitive stops flow execution and throws an exception.

1. In the palette, expand **Error Handling** and select **Fail**.
2. Click an empty portion of the flow diagram to the right of the **TransformResponse** primitive.
3. With the fail primitive selected, switch to the **Description** tab in the **Properties** view.
4. Change the **Display Name** to: `TransformResponseFail`



5. Right-click the **fail** terminal on the **TransformResponse** primitive and click **Add Connection** from the menu.
6. Click the **in** terminal on the **TransformResponseFail** primitive to add the connection.
7. To aid readability, right-click the flow diagram and click **Layout Contents** from the menu.

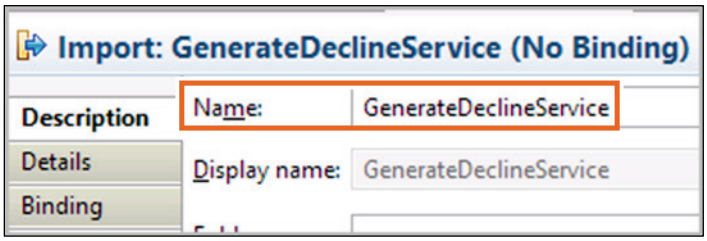
The response flow diagram resembles the following figure:



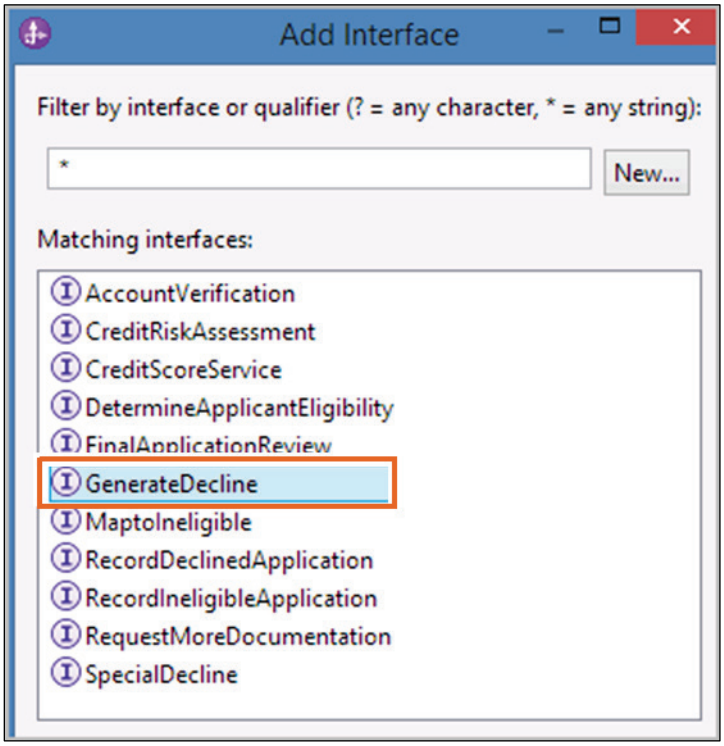
8. Save your changes.
   Verify that the **Problems** view contains no errors. You can ignore any warnings.
9. Close the mediation flow editor.

19. Add an import component with an SCA binding that is named `GenerateDeclineService` to the RouterMediationService assembly diagram. The GenerateDeclineService import uses the GenerateDecline interface. The target of the import is the GenerateDeclineExport in the FoundationServices module.

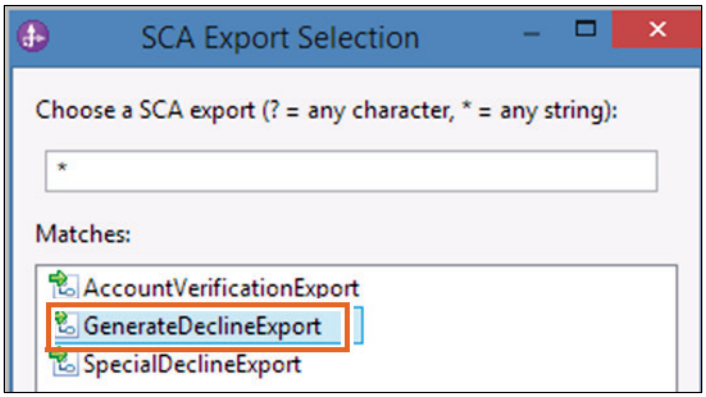
1. In the **RouterMediationService** assembly diagram, expand the **Components** section of the palette and click **Import**.
2. Click any blank space on the canvas to add the import component.
3. Switch to the **Description** tab in the **Properties** view.
4. Change the **Name** to: `GenerateDeclineService`



5. Right-click **GenerateDeclineService** and click **Add Interface** from the menu.
6. In the Add Interface dialog box, select **GenerateDecline**.

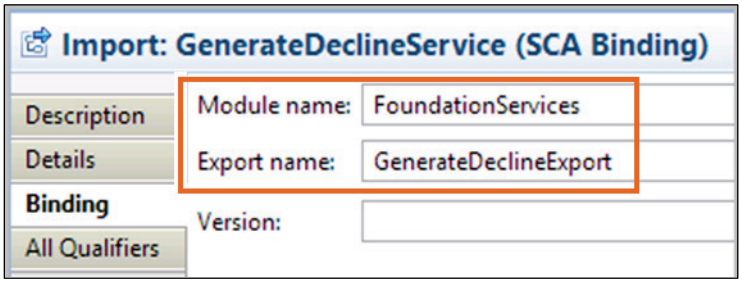


7. Click **OK**.
8. Right-click the **GenerateDeclineService** import and click **Generate Binding > SCA Binding**.
9. Switch to the **Binding** tab in the **Properties** view.
10. For **Export Name**, click **Browse** and select **GenerateDeclineExport** in the SCA Export Selection dialog box.

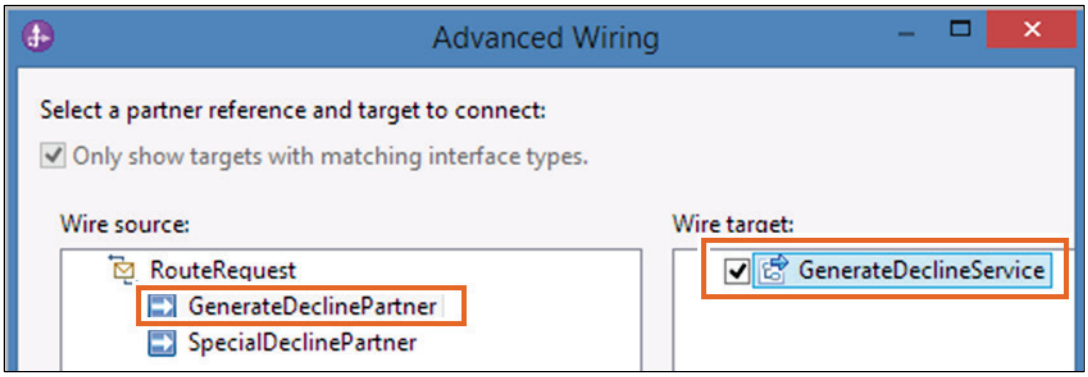


11. Click **OK**.

    Your **Binding** tab resembles the following figure:



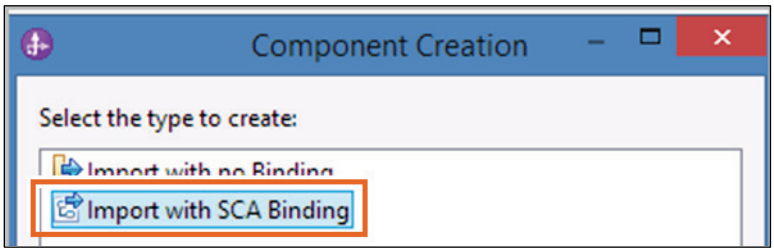
12. Right-click the **RouteRequest** component and click **Wire (Advanced)** from the menu.
13. In the Advanced Wiring dialog box, in the “Wire source” window, select GenerateDeclinePartner**.**
14. In the “Wire target” window, select **GenerateDeclineService**.



15. Click **OK**.
16. Save your changes.

    You can verify the wiring by hovering over the reference. The dialog box lists: `GenerateDeclinePartner`

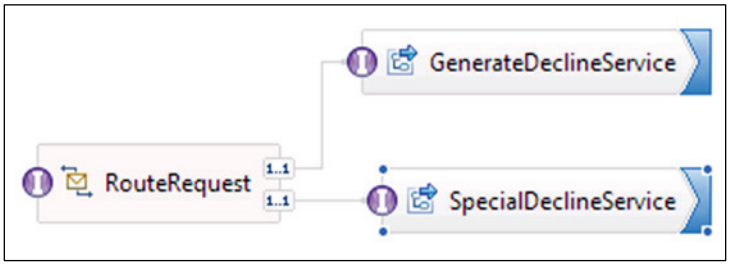
20. Add an import component with an SCA binding that is named `SpecialDeclineService` to the RouterMediationService assembly diagram. The SpecialDeclineService import uses the SpecialDecline interface. The target of the import is the SpecialDeclineExport in the FoundationServices module.
    1. In the Business Integration view, expand **FoundationServices > Assembly Diagram**.
    2. Drag **SpecialDeclineExport** onto the **RouterMediationService** assembly diagram.
    3. In the Component Creation dialog box, select **Import with SCA Binding**.



    4. Click **OK**.
    5. Switch to the **Description** tab in the **Properties** view.
    6. Change the **Name** to: `SpecialDeclineService`

    7. Right-click the **SpecialDeclineService** component and click **Wire to Existing** from the menu.
    8. Save your changes.

       You can verify the wiring by hovering over the reference. The dialog box lists: `SpecialDeclinePartner`. Your assembly diagram resembles the following figure:



21. Create an export that is named RouteRequestExport, with SCA binding, for the RouteRequest mediation component.

1. Right-click RouteRequest and click Generate export > SCA binding.
2. Accept the default export name: `RouteRequestExport`.

   Your assembly diagram resembles the following figure:

3. Save and close the assembly diagram. The **Problems** view contains no errors. You can ignore any warnings.

Part 3. Test a mediation module that contains a message filter mediation primitive and a Mapping primitive.

In this exercise, you implemented a mediation module that is named `RouterMediationService`. The AccountVerification business process invokes this service when an application is declined during FinalApplicationReview. In this portion of the exercise, you wire the RouterMediationService to the AccountVerification process component on the FoundationModule assembly diagram.

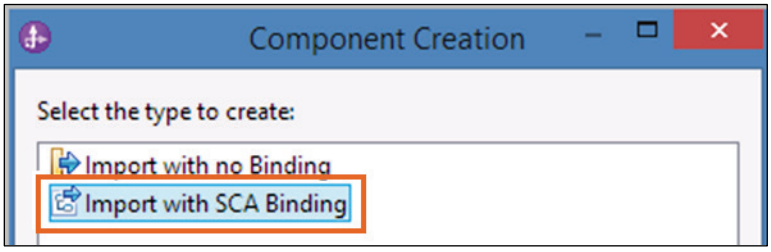
To assemble the applications:

1. Create an import with SCA binding on the FoundationModule assembly diagram.

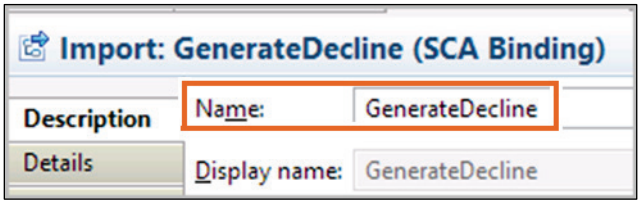
The import component is named `GenerateDecline`. The target of the import is the RouteRequestExport component in the RouterMediationService module.

1. In the Business Integration view, expand **FoundationModule** and double-click **Assembly Diagram.**

2. In the Business Integration view, expand **RouterMediationService > Assembly Diagram**.
3. Drag **RouteRequestExport** from **RouterMediationService** onto the **FoundationModule** assembly diagram.
4. In the Component Creation dialog box, select **Import with SCA Binding**.



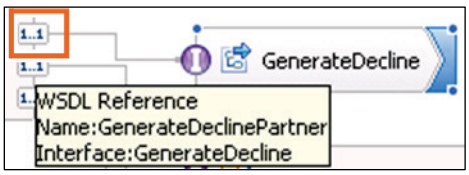
5. Click **OK**.
6. Switch to the **Description** tab in the **Properties** view.
7. Change the **Name** to: `GenerateDecline`



2. Wire the GenerateDeclinePartner reference on the AccountVerification process component to the newly created GenerateDecline import.

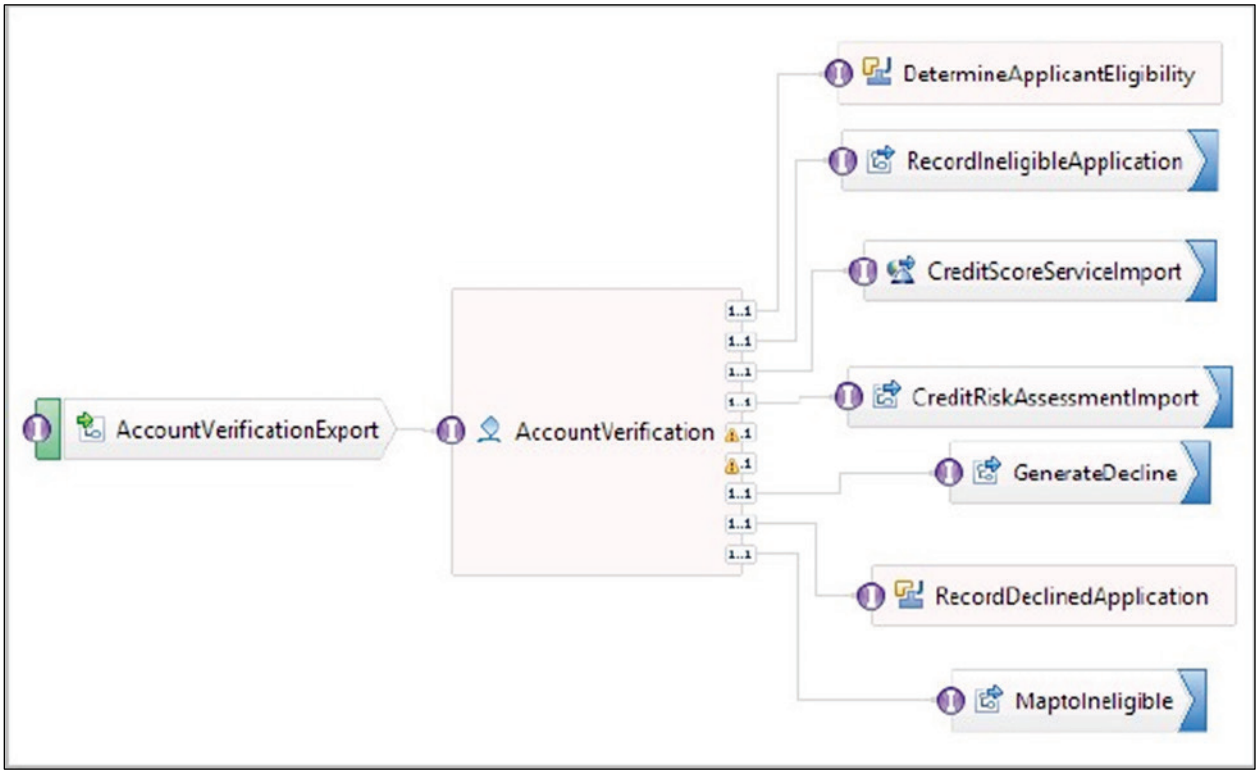
1. Right-click **GenerateDecline** and click **Wire to Existing** from the menu.
2. Verify the wiring by hovering over the reference component that is wired to **GenerateDecline**.

   The dialog box lists: `GenerateDeclinePartner`



3. Press Ctrl+S to save the **FoundationModule** assembly diagram.

   Your assembly diagram resembles the following figure:



4. Close the assembly editor.

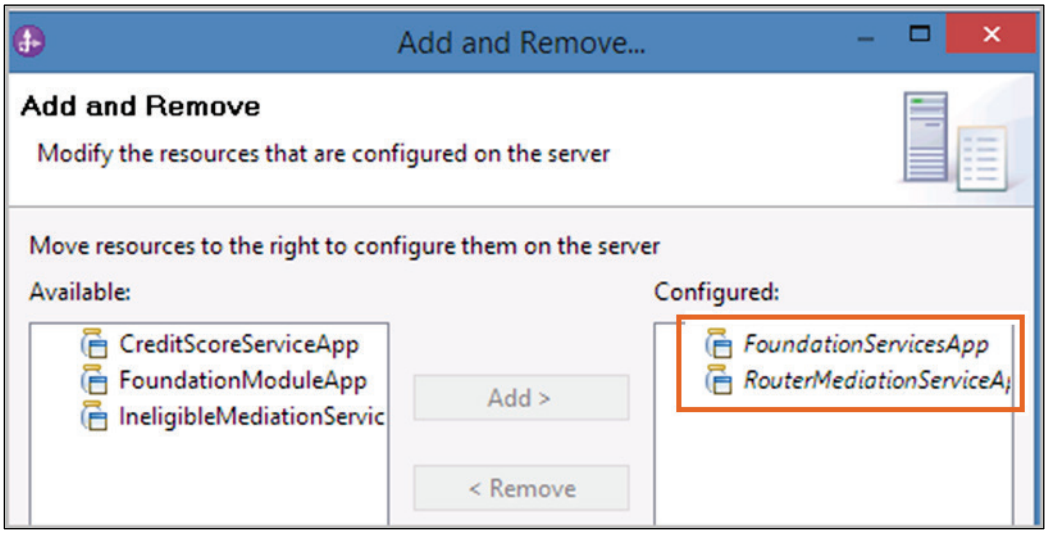
Testing the mediation module

In this portion of the exercise, you test the RouterMediationService mediation module. When an application passes through FinalApplicationReview, the business process follows either the “approved” path or the “declined” path, which depends on the value in applicationDecision.

If applicationDecision is `true` (the application was approved), the process proceeds to the AccountVerificationReply activity. If applicationDecision is `false` (the application was declined), the process proceeds to the GenerateDecline activity. GenerateDecline processes the declined application and bases its action on the value in the creditRisk attribute. If creditRisk is `MED`, then SpecialDeclineService is called. If creditRisk is `HIGH`, then GenerateDeclineService is called.

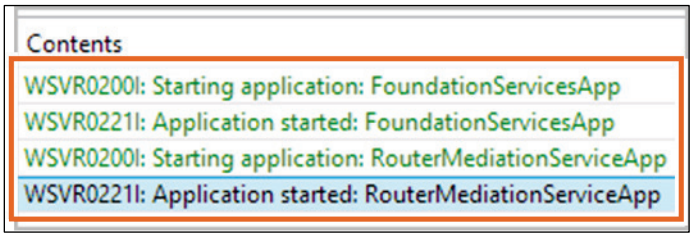
To test the **RouterMediationService** mediation module:

1. Start the server (if it is not already running) and deploy RouterMediationServiceApp and FoundationServicesApp.
   1. Start the process server if stopped, by double-clicking the desktop shortcut.
   2. In the **Servers** view, right-click **IBM Process Server v8.6 at localhost** and click **Add and Remove** from the menu.
   3. In the **Available** list, double-click **RouterMediationServiceApp** and **FoundationServicesApp** to add them to the **Configured** list.



   4. Click **Finish**.

5. When the applications are deployed and started, the messages `Application started: RouterMediationServiceApp` and `Application started: FoundationServicesApp` are listed in the **Server Logs** view.

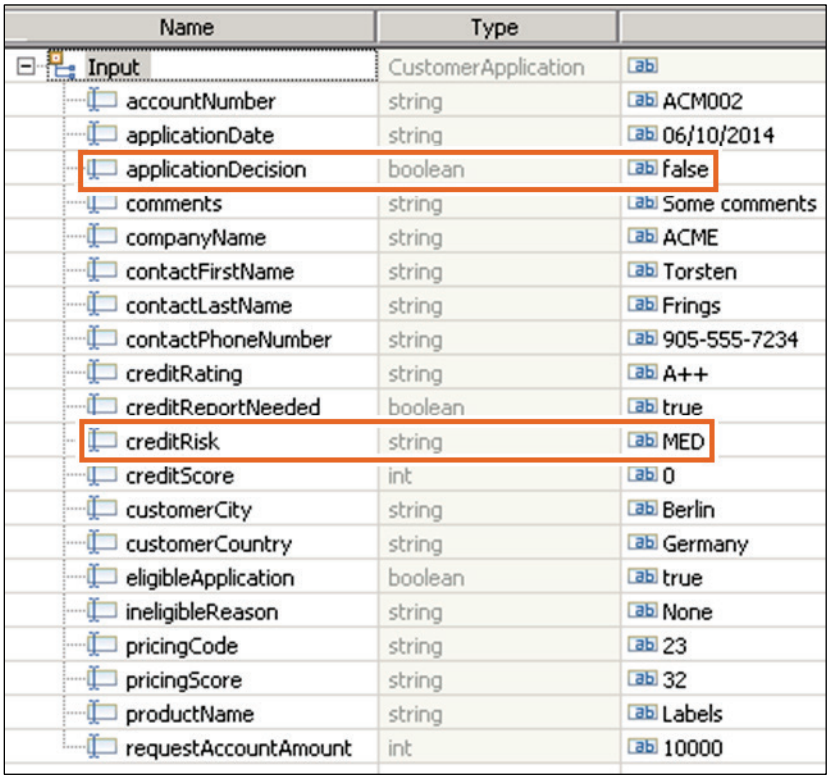


6. If any module has a **Stopped** status, then right-click the module and click **Restart** from the menu. If prompted to do so, republish the module. Wait for the server status to change to **Started, Synchronized**. If the server has a status of **Started, Publishing**, then clicking the server refreshes the status to **Started, Synchronized**.

2. Test the RouteRequest flow component by invoking the RouteRequestExport.

1. In the Business Integration view, expand **RouterMediationService** and double-click **Assembly Diagram**.
2. Right-click **RouteRequestExport** and click **Test Component** from the menu.
3. When the integrated test client opens, in the **Initial request parameters** section, right-click **Input**, click **Import from File** from the menu, select **C:\labfiles\Support Files\Ex11\EX11_Test_Data.xml**, and click **Open**.

The input parameters are populated with the required test data. The value for applicationDecision is `false` and creditRisk is `MED`. They test the SpecialDeclineService.



Name	Type	
Input	CustomerApplication	
accountNumber	string	ACM002
applicationDate	string	06/10/2014
applicationDecision	boolean	false
comments	string	Some comments
companyName	string	ACME
contactFirstName	string	Torsten
contactLastName	string	Frings
contactPhoneNumber	string	905-555-7234
creditRating	string	A++
creditReportNeeded	boolean	true
creditRisk	string	MED
creditScore	int	0
customerCity	string	Berlin
customerCountry	string	Germany
eligibleApplication	boolean	true
ineligibleReason	string	None
pricingCode	string	23
pricingScore	string	32
productName	string	Labels
requestAccountAmount	int	10000

4. Alternatively, enter the following test data in the **Initial request parameters** section (verify that the data matches the XML file):
    - accountNumber: `ACM002`
    - applicationDate: `06/10/2014`
    - applicationDecision: `false`
    - comments: `Some comments`
    - companyName: `ACME`
    - contactFirstName: `Torsten`
    - contactLastName: `Frings`
    - contactPhoneNumber: `905-555-7234`
    - creditRating: `A++`
    - creditReportNeeded: `true`
    - creditRisk: `MED`
    - creditScore: `0`
    - customerCity: `Berlin`
    - customerCountry: `Germany`
    - eligibleApplication: `true`
    - ineligibleReason: `None`
    - pricingCode: `23`
    - pricingScore: `32`
    - productName: `Labels`
    - requestAccountAmount: `10000`

5. Click the **Continue** icon to start the test thread.
6. In the “Select a Deployment Location” dialog box, select **IBM Process Server v8.6 at localhost** and click **Finish**.
7. In the User Login dialog box, accept the default entries for **User ID** and **Password** and click **OK**.

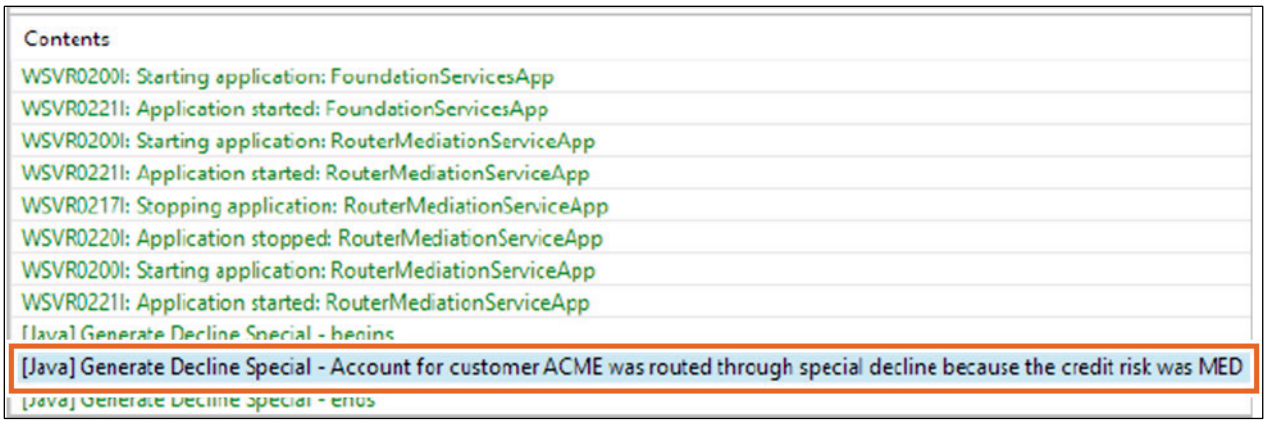
   These fields are set to `admin` and `web1sphere` by default.

   The test run is complete when the blue, square stop node is displayed in the Events window.

   To examine the test trace and the log messages, you will switch to the **Server Logs** view.

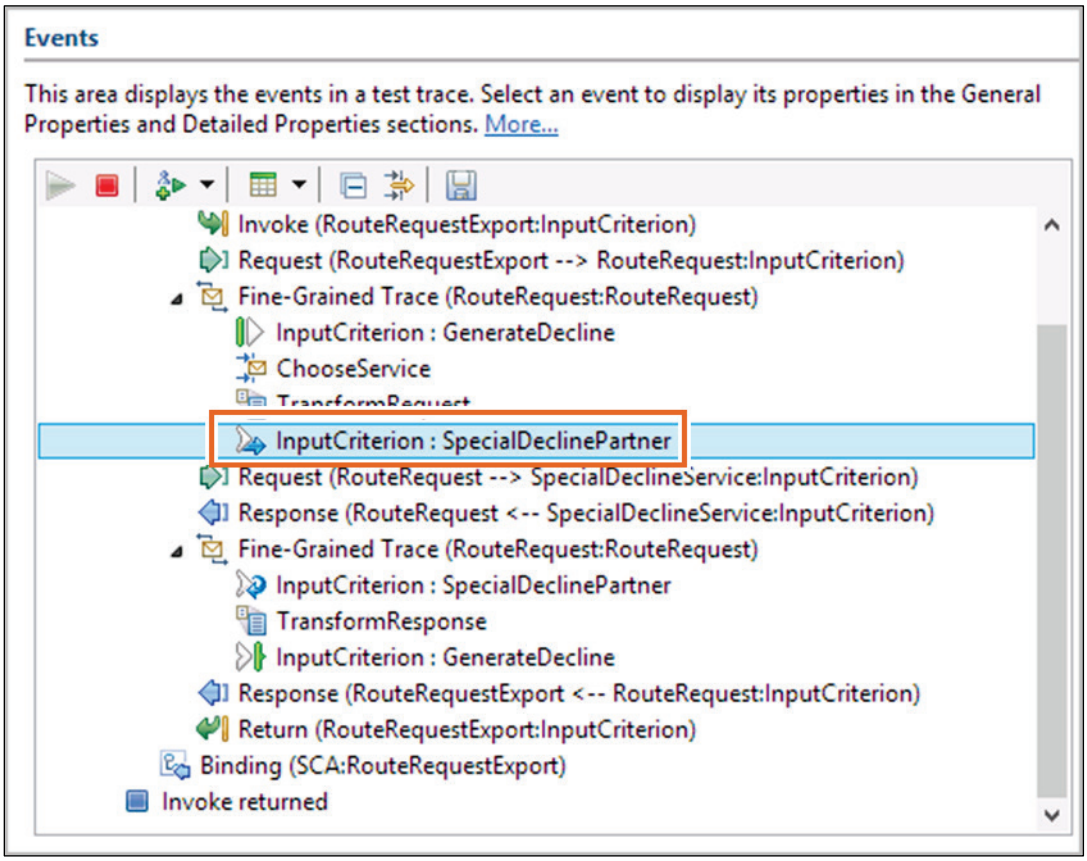
   The following messages are displayed in the **Server Logs** view. They indicate that the application was sent to the special decline service.

```
Generate Decline Special - begins
Generate Decline Special - Account for customer ACME was routed
through special decline because the credit risk was MED
Generate Decline Special - ends
```



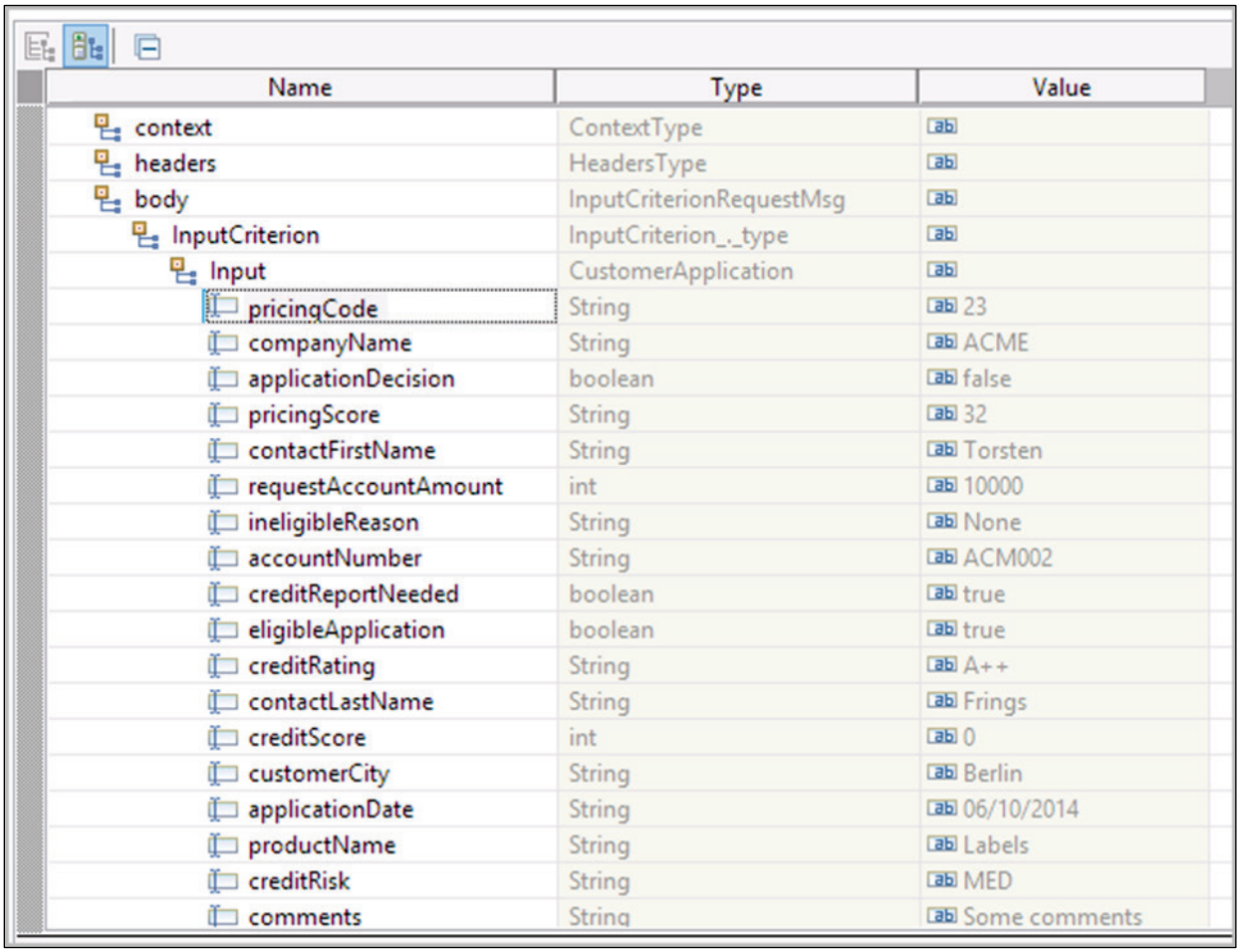
8. In the Events window of the test client, select the first instance of the **InputCriterion : SpecialDeclinePartner** callout event.

   The **SpecialDeclinePartner** service is invoked when **applicationDecision** is `false` and **creditRisk** is `MED`.

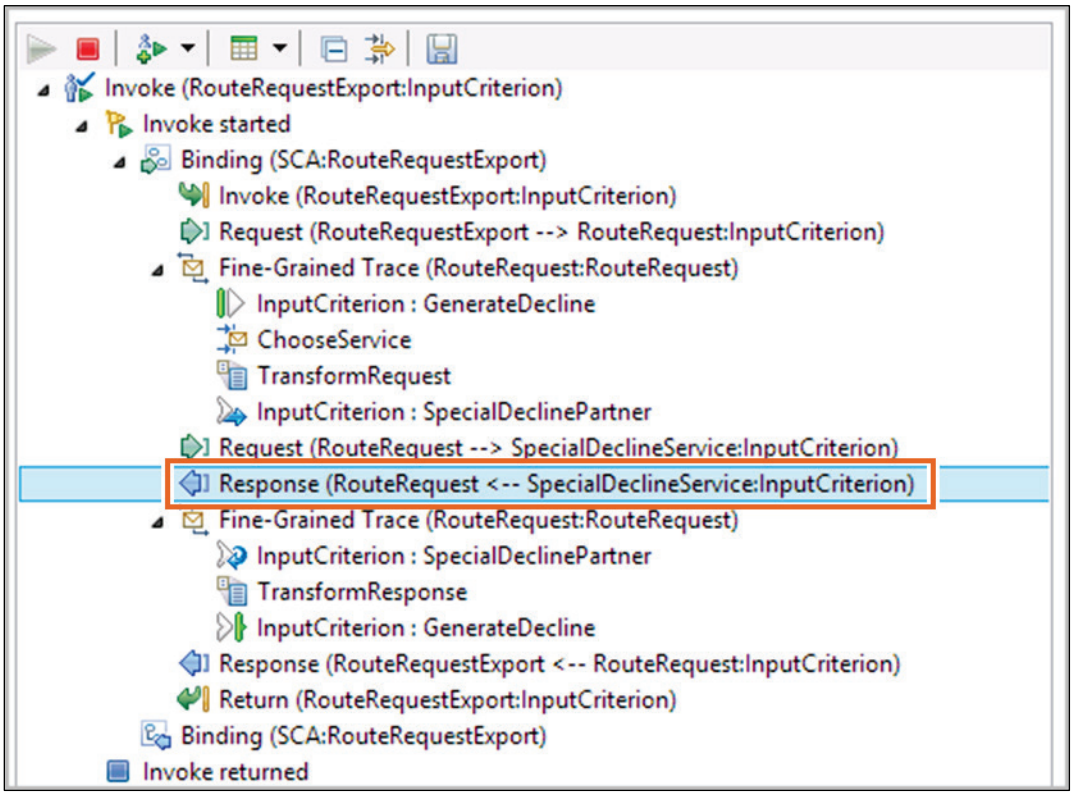


9. In the **Mediation Message** section, examine the data in the **body** of the message.

   You might find it necessary to collapse the headers section to see the data. The XML map successfully transformed the message by moving all of the attribute values.

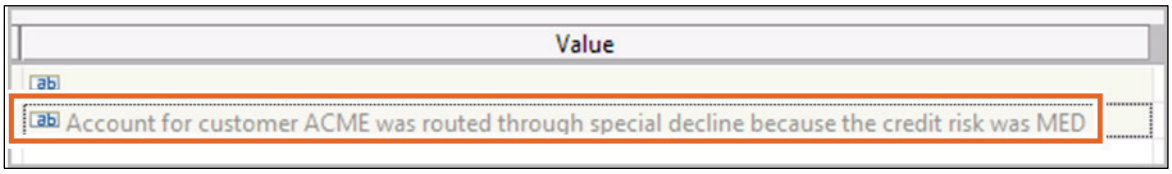


10. In the Events window, select **Response (RouteRequest <-- SpecialDeclineService:InputCriterion)**.



11. In the **Response parameters** section, examine the output message.

    You can resize the columns to view the message. The message reads: `Account for customer ACME was routed through special decline because the credit risk was MED`

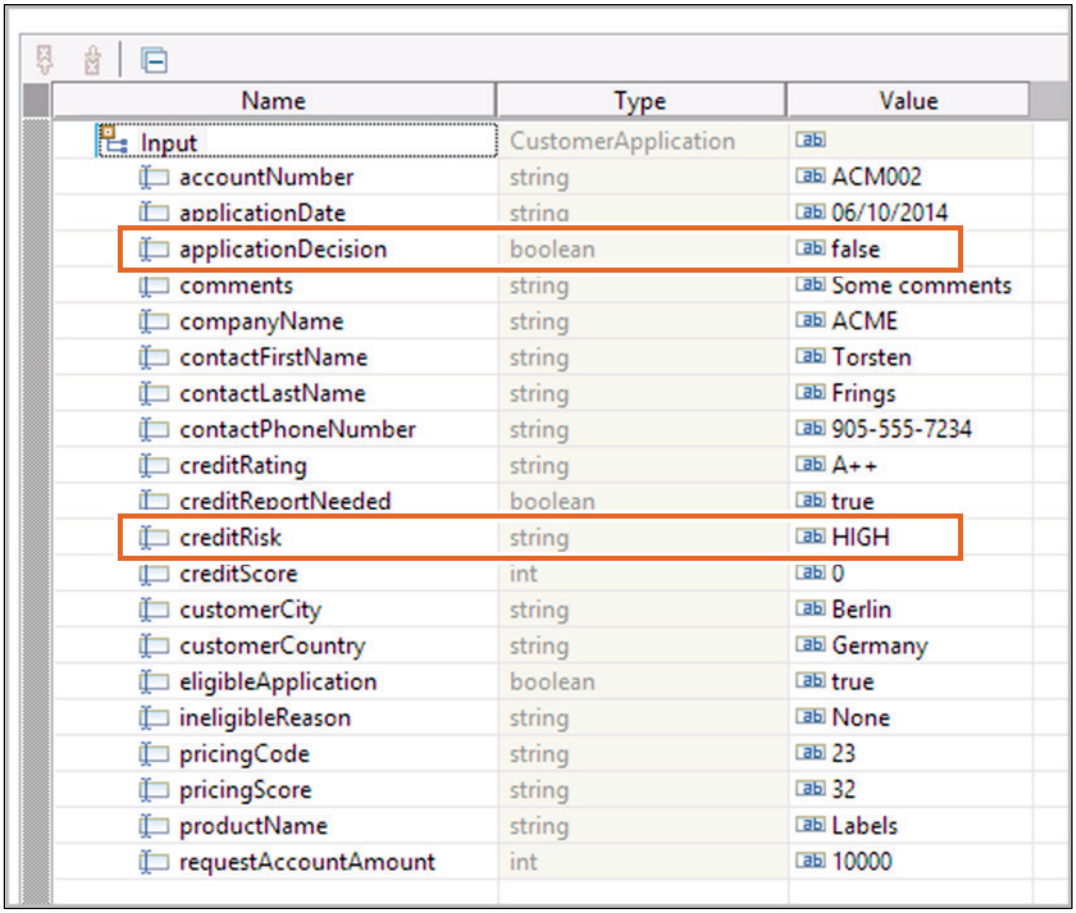


3. Invoke the test a second time. Test the GenerateDeclineService path by setting applicationDecision to false and creditRisk to HIGH.

1. In the Events window, click the **Invoke** icon on the action bar.

2. In the **Initial request parameters** section, right-click **Input**, click **Import from File** from the menu, select **C:\labfiles\Support Files\Ex11\EX11_Test_Data2.xml**, and click **Open**.

   The input parameters are populated with the required test data. The value for **applicationDecision** is `false` and **creditRisk** is `HIGH`. They test the GenerateDeclineService.



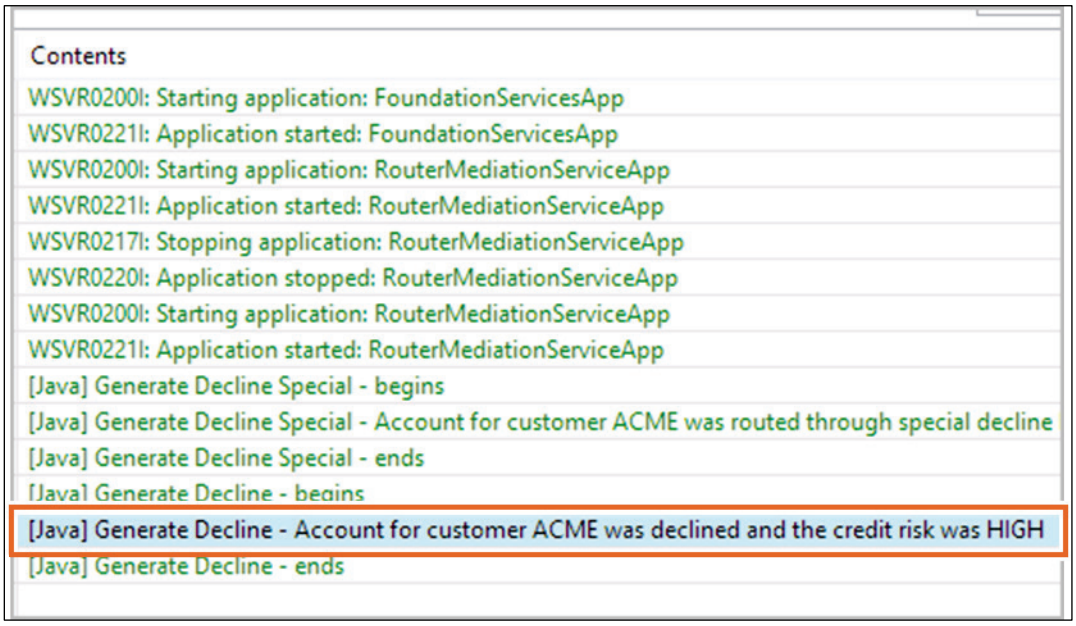
3. Alternatively, enter the following test data in the **Initial request parameters** section:
   - accountNumber: `ACM002`
   - applicationDate: `06/10/2014`
   - applicationDecision: `false`
   - comments: `Some comments`
   - companyName: `ACME`
   - contactFirstName: `Torsten`
   - contactLastName: `Frings`
   - contactPhoneNumber: `905-555-7234`
   - creditRating: `A++`
   - creditReportNeeded: `true`
   - creditRisk: `HIGH`
   - creditScore: `0`
   - customerCity: `Berlin`
   - customerCountry: `Germany`
   - eligibleApplication: `true`
   - ineligibleReason: `None`
   - pricingCode: `23`
   - pricingScore: `32`
   - productName: `Labels`
   - requestAccountAmount: `10000`
4. In the Events window, click the **Continue** icon in the action bar.
   The test run is complete when the blue, square stop node is displayed in the Events window.

4. Examine the test trace and the log messages.

1. Switch to the **Server Logs** view.

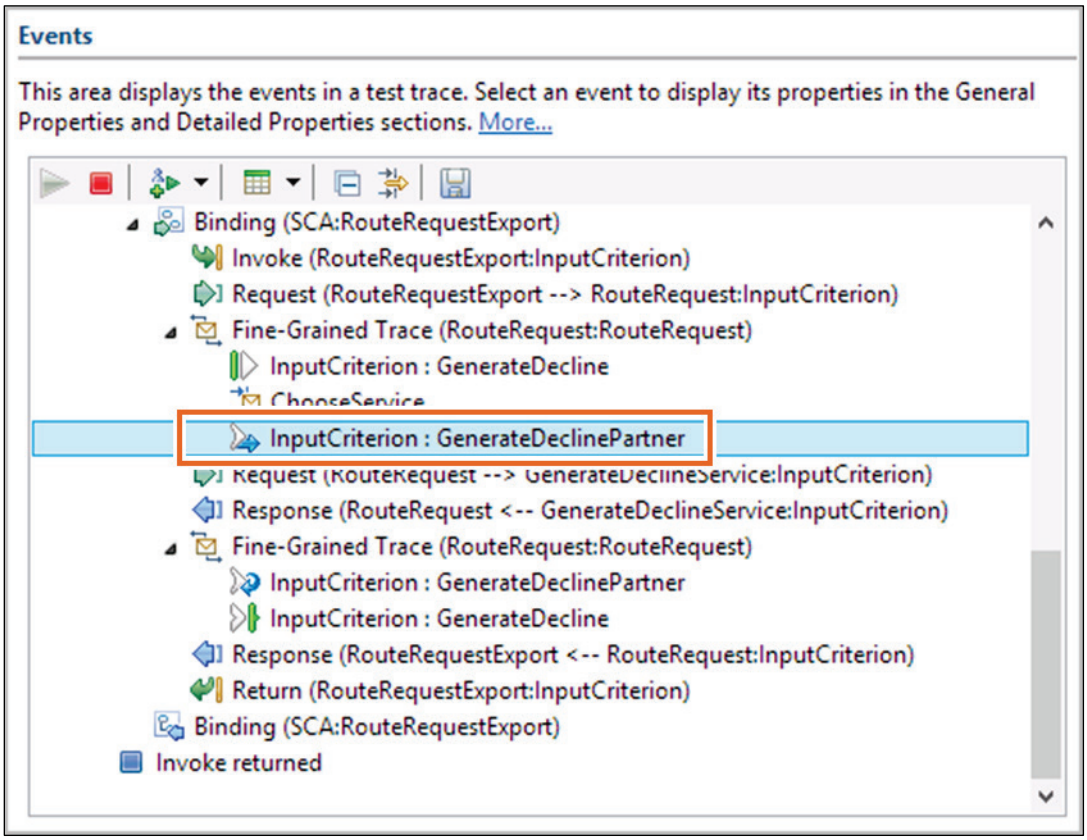
The following messages are displayed in the **Server Logs** view. They indicate that the application was sent to the special decline service:

```
Generate Decline - begins
Generate Decline Special - Account for customer ACME was declined
and the credit risk was HIGH
Generate Decline - ends
```



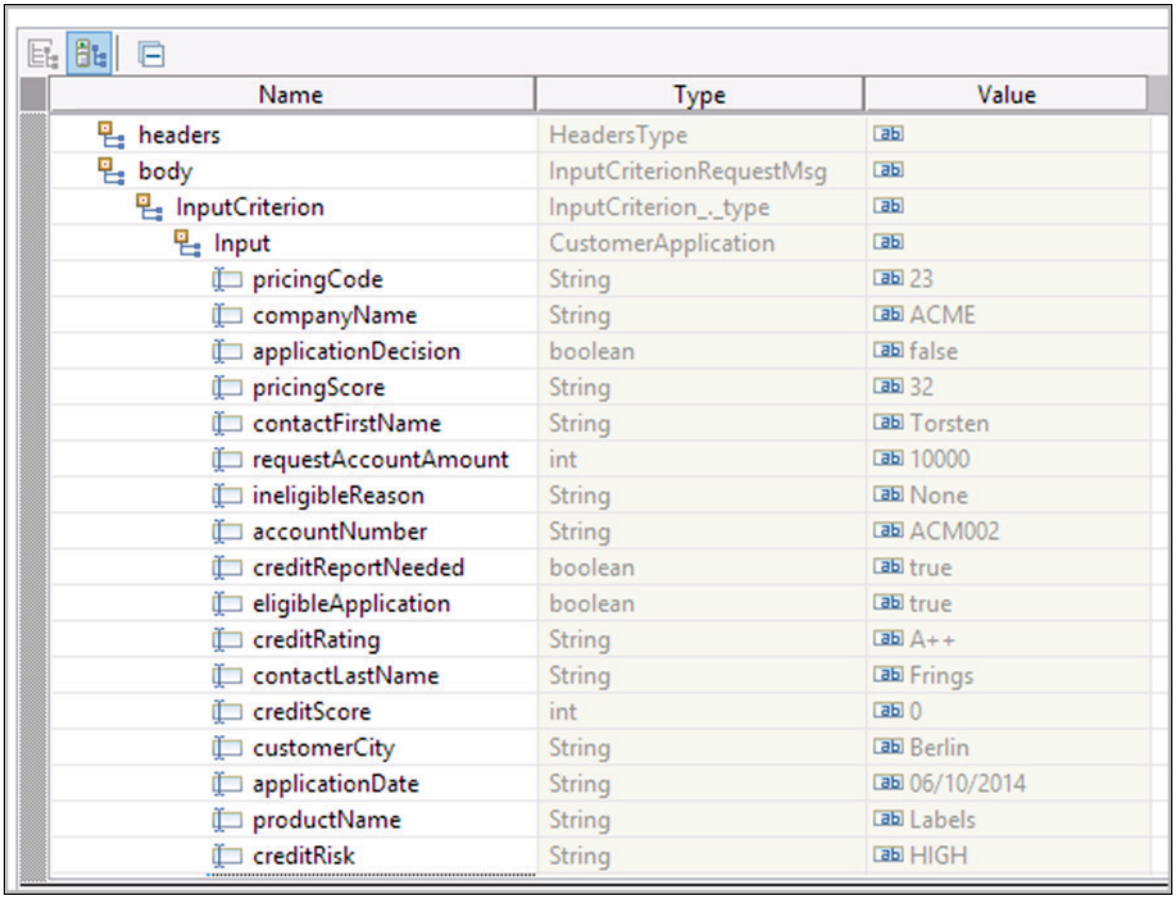
2. In the Events window of the test client, select the **InputCriterion: GenerateDeclinePartner** callout event.

   The GenerateDeclinePartner service is invoked when applicationDecision is `false` and creditRisk is `HIGH`.



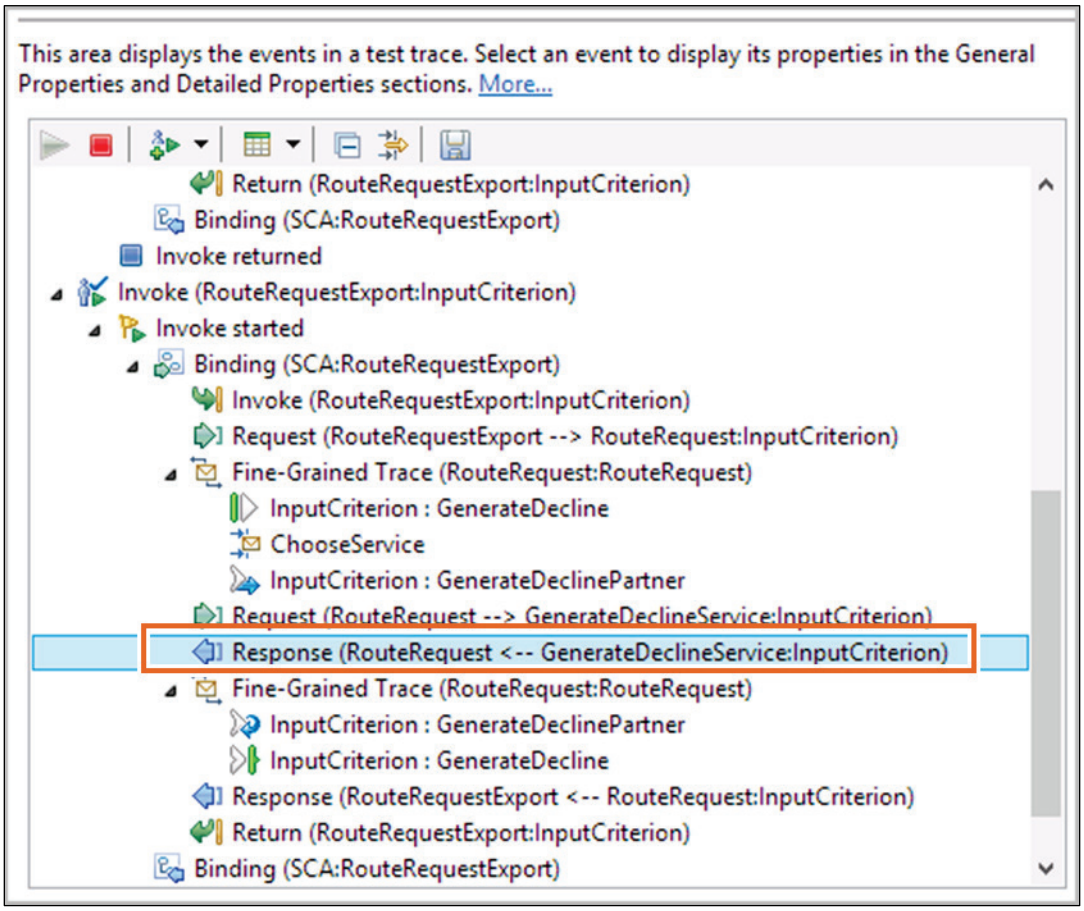
3. In the **Mediation Message** section, examine the data in the body of the message.

   You can collapse the headers section to see the data. The XML map successfully transformed the message by moving all of the attribute values.



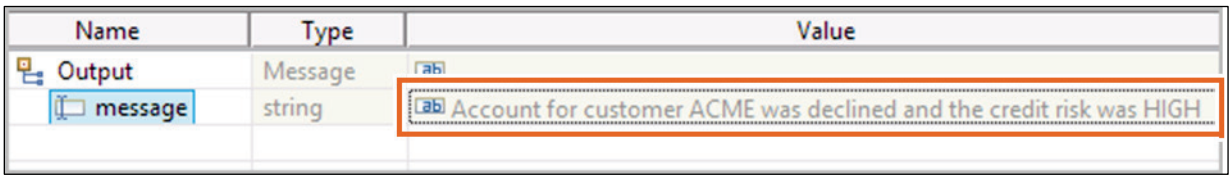
Name	Type	Value
headers	HeadersType	
body	InputCriterionRequestMsg	
InputCriterion	InputCriterion_._type	
Input	CustomerApplication	
pricingCode	String	23
companyName	String	ACME
applicationDecision	boolean	false
pricingScore	String	32
contactFirstName	String	Torsten
requestAccountAmount	int	10000
ineligibleReason	String	None
accountNumber	String	ACM002
creditReportNeeded	boolean	true
eligibleApplication	boolean	true
creditRating	String	A++
contactLastName	String	Frings
creditScore	int	0
customerCity	String	Berlin
applicationDate	String	06/10/2014
productName	String	Labels
creditRisk	String	HIGH

4. In the Events window, select **Response (RouteRequest <-- GenerateDeclineService:InputCriterion)**.



5. In the **Response parameters** section, examine the output message.

   The message reads: Account for customer ACME was declined and the credit risk was HIGH.



6. Close the **RouterMediationService_test** tab and click **No** when you are prompted to save the test trace.

5. Remove all of the projects and stop the server.

1. In the Servers view, right-click IBM Process Server v8.6 at localhost and click Add and Remove from the menu.

2. Click **Remove All**.
3. Click **Finish**.
4. Close IBM Integration Designer.

Results
In this exercise, you created a mediation module that contained a message filter primitive and a Mapping primitive and then tested the module in the IBM Integration Designer test environment.

IBM Training

Business Space

IBM Business Process Manager V8.6

IBM Training

Unit objectives

- Describe the purpose and business value of Business Space
- Describe the types of widgets that are available in Business Space
- Describe the types of templates that are available in Business Space
- Explain how to create and configure a new space in Business Space

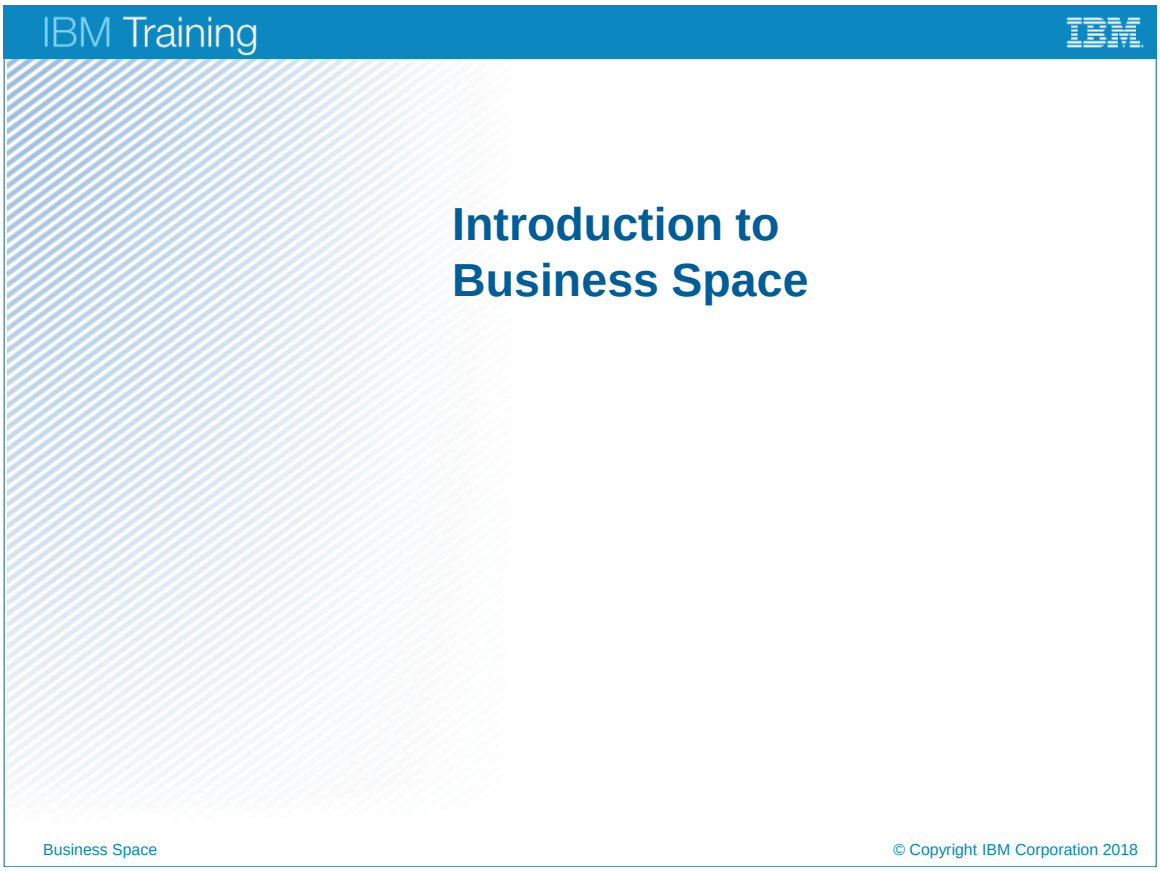
Unit objectives

IBM Training

Topics

- Introduction to Business Space
- Business Space terminology
- Configure a new space

Topics



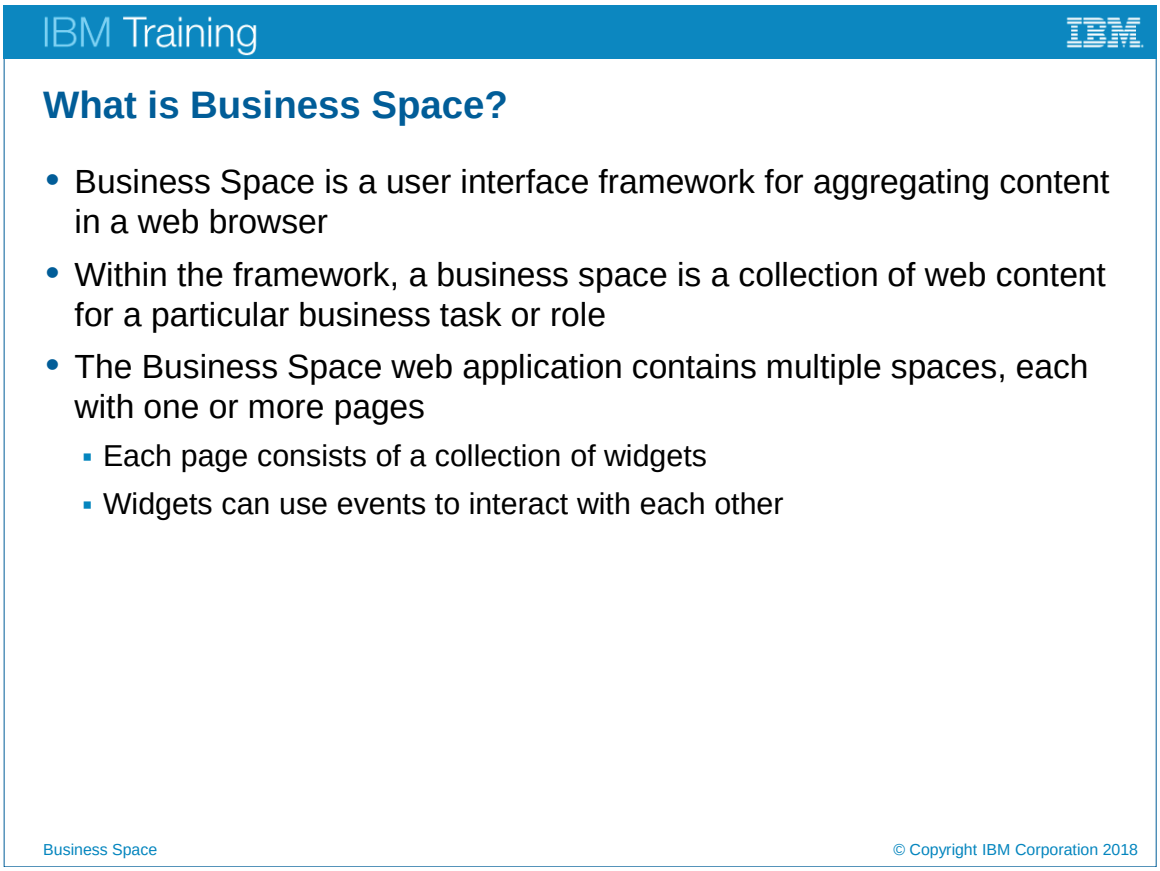
Introduction to Business Space

IBM Training

Problem statement that Business Space addresses

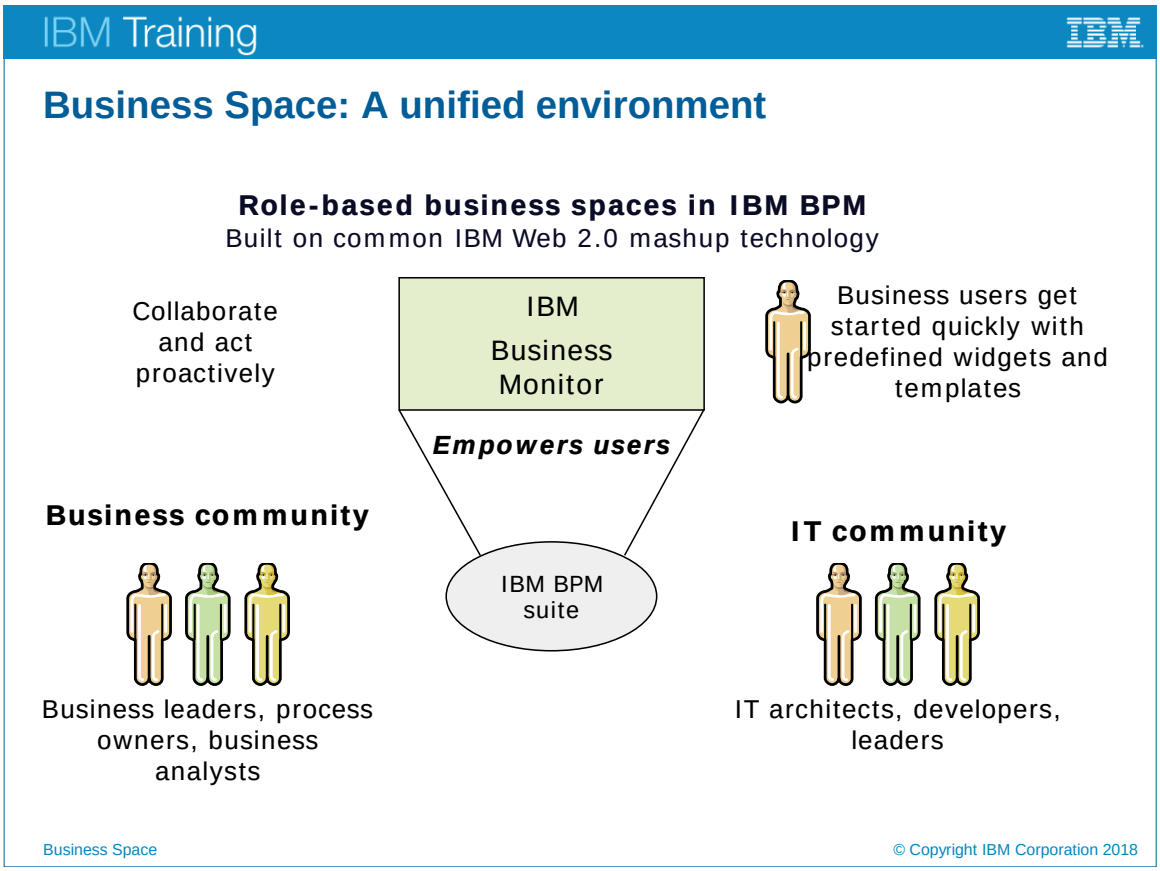
- A need for a common front end for products in the business process management portfolio
- Create a front end for users of IBM Business Process Manager that is not product-centric
- Move away from a product-centric solution to a user solution
- Bring users of different roles to a page to solve a problem
- Make Business Space the entry point into the business process management product portfolio

Problem statement that Business Space addresses



What is Business Space?

Business Space is a ready-to-use Web 2.0 IBM BPM client for business users. It is a collection of related web content that provides you with insight into your business and the capability to react to changes within it. Business Space unifies the IBM Business Process Manager user interface space for business users. It allows users to use a single window when working with all of the business activities that take place in different products in the IBM Business Process Manager suite. Business Space is a mashup of IBM Business Process Manager widgets that are targeted for a business user.



Business Space: A unified environment

The business space provides a rich, holistic, and transformative experience to users across the business process management and connectivity product portfolio stack. The business space is a common infrastructure component that comes with all business process management products.

It empowers the business user by enabling key processes (for example, dashboards, point-of-business agility, and composite application policy).

The business space integrates the creation of the user experience with the authoring of the business application itself:

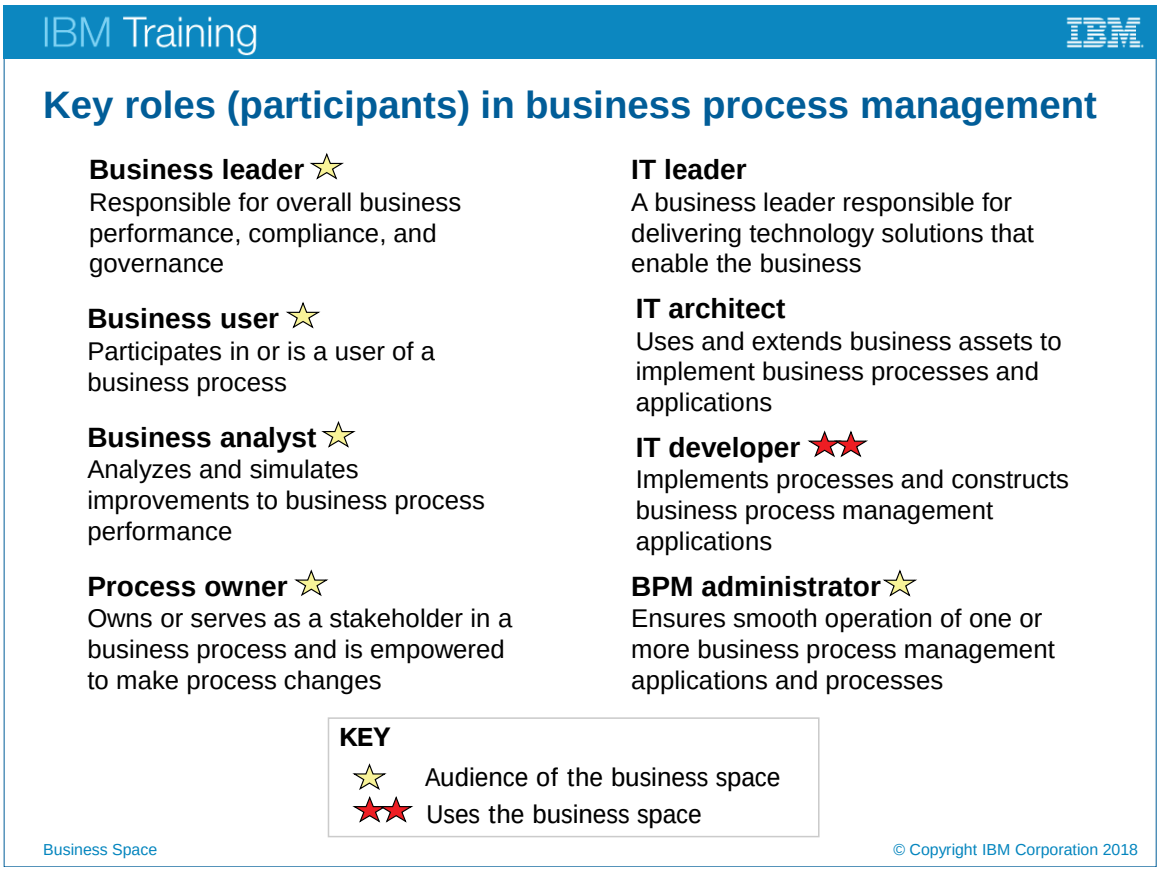
- It allows both business users and IT developers to create and deliver rich content for a broader audience of BPM users across a range of roles.
- It uses the same capabilities for empowering the user experience that is built for the business application.

It also allows collaboration between all participants in the BPM process:

- It provides different levels of collaboration in day-to-day activities across role boundaries: human tasks and coordination, strategy, review, modeling of processes, and organizational navigation.

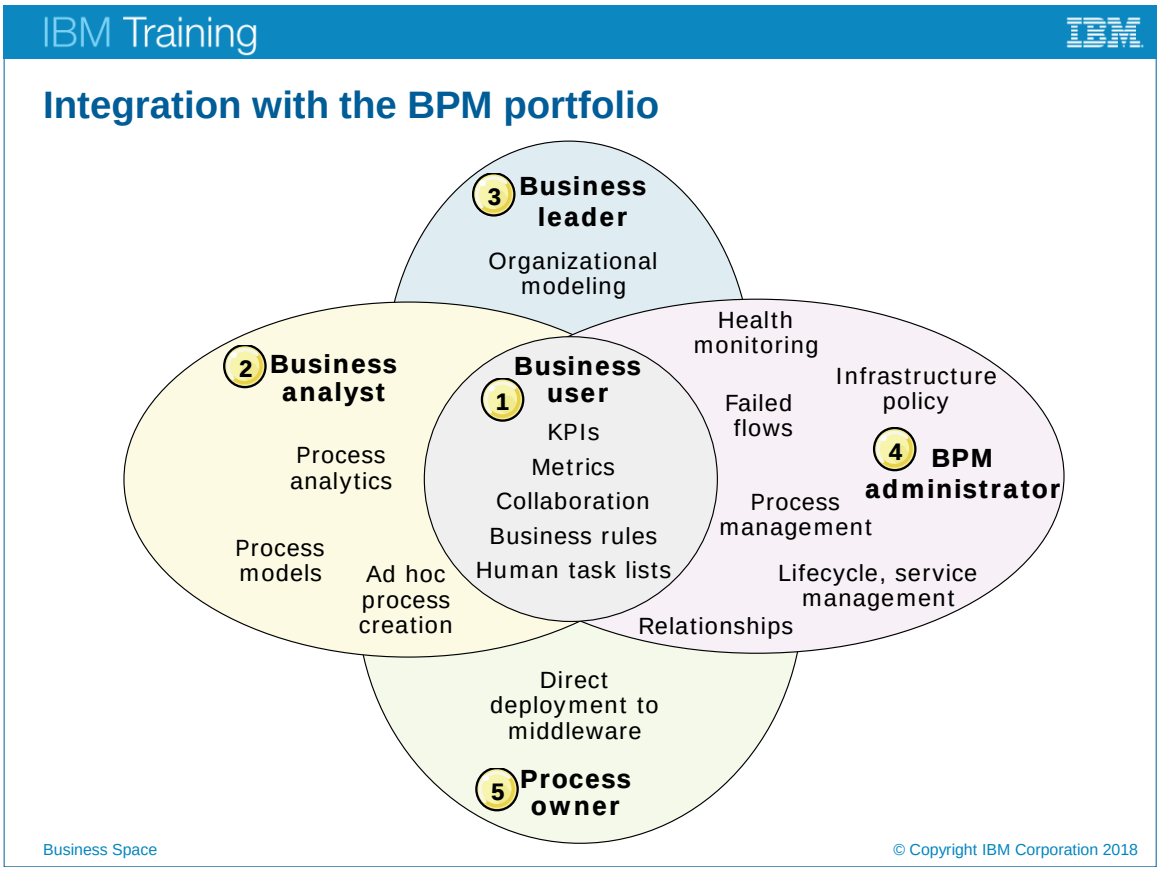
It transforms IT administration into a business-centric and solution-centric experience:

- It shifts the focus of the user experience away from deployment-centric to business application-centric administration and integrates IT administration into the BPM lifecycle optimization process.



Key roles (participants) in business process management

The technology behind the business space provides a Web 2.0 user with a view of IBM BPM data for a business user. The technology provides a holistic view into how business users can handle their day-to-day interaction with IBM BPM data. The overriding concept with the business space is to provide content and solutions from a common front end.



Integration with the BPM portfolio

The business space is developed to integrate the Business Process Management portfolio around five scenarios:

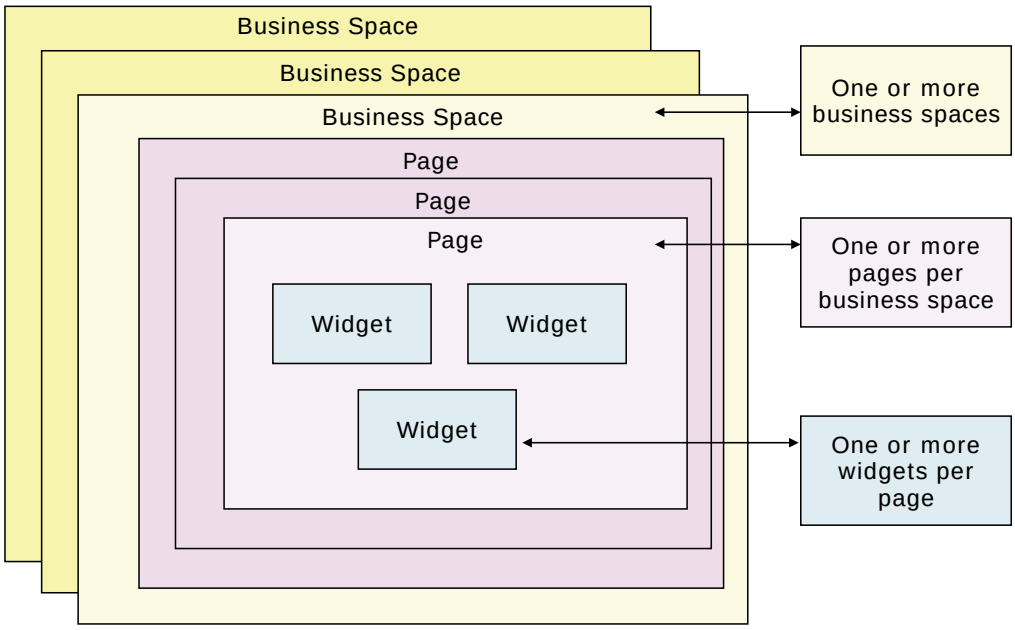
1. The business space for managing tasks and human workflow: This space helps business users to process tasks efficiently and collaborate to get the job done.
2. The business space for initiating process improvement: This space helps change business performance through analysis, simulation, and refinement of business process decisions.
3. The business space for managing business performance: This space is used to harvest information about the business to make better business decisions and coordinate execution.
4. The business space for solution management: Ensures that business applications keep running optimally and that they support the business.
5. Generating a business space to experiment with the business processes: This scenario provides business agility through automation and direct refinement of business processes.

Business Space terminology

IBM Training

Business Space hierarchy

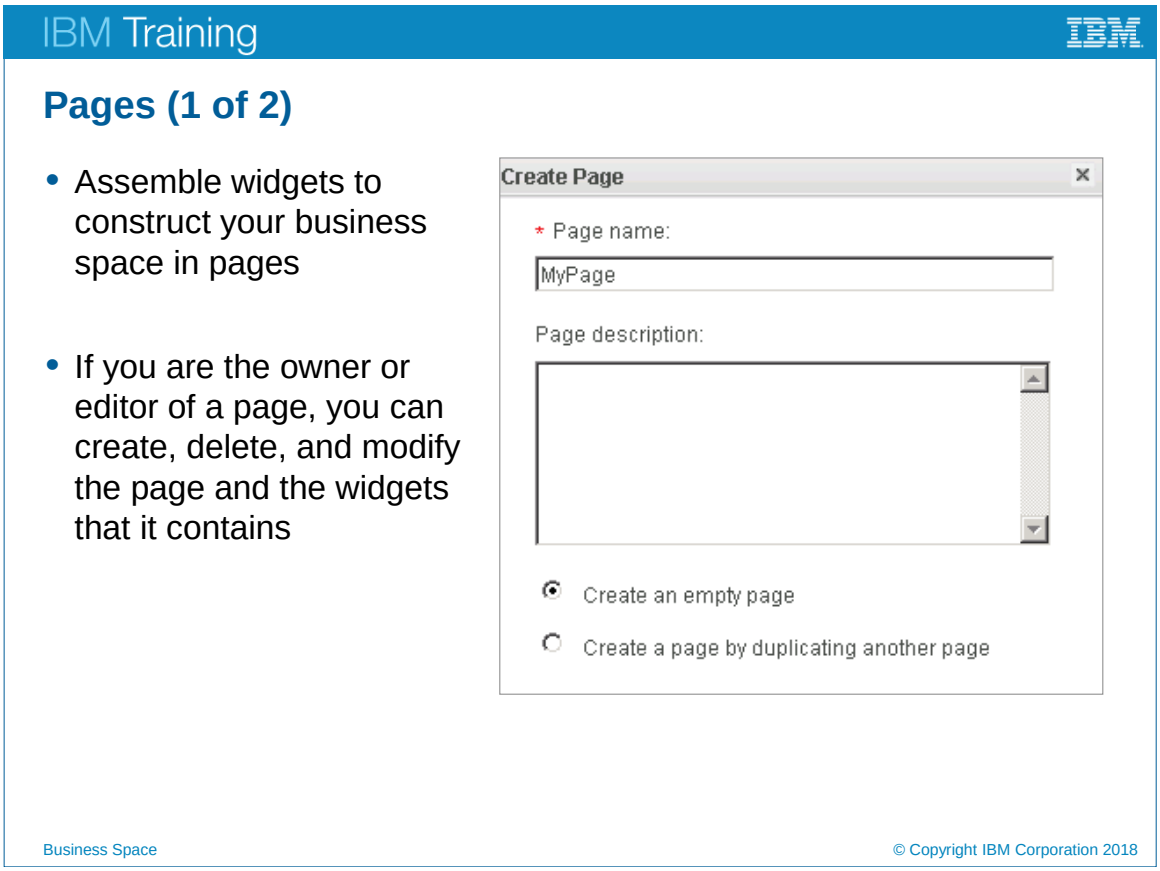
- A business space is a web page that has several components:



Business Space

Business Space hierarchy

The Business Space viewing web page consists of one or more business spaces. Each business space can contain one or more tabbed pages, and each page can contain one or more widgets. As a business user, you control how many business spaces you want on your Business Space web page. Also, for each business space, you control how many pages and widgets you would like to see. Templates are provided for specific business spaces that are based on the specific business functions, which are based on different roles of a business user.



Pages

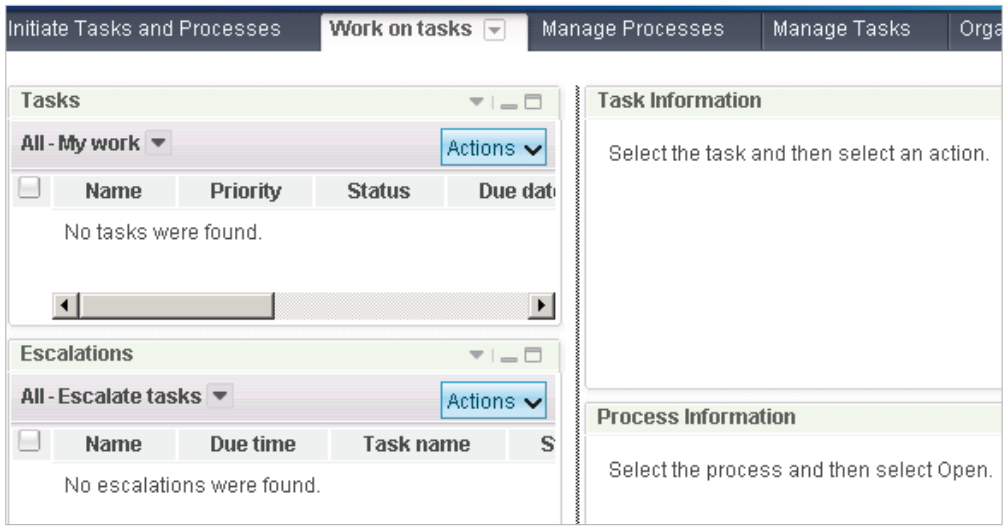
Pages are the medium in which you assemble widgets to construct your business space. If you are the owner or editor of a page, you can create, delete, and modify the page and the widgets that it contains.

- **Creating pages**: If you can edit a business space, you can create one or more pages in it. You can then place widgets on the pages. You can import a page into a business space that you own or can edit. Importing is a way of creating a page that is already populated with configured widgets. You can export a page from a business space. Exporting a page is useful if you want to give someone a copy of the page without giving them access to it or if you want to re-create the page later.
- **Switching pages**: If you are viewing a page in a business space, you can switch to a different page. When you switch pages, Business Space stores which page you last went to in the most recent nine spaces. When you return to one of these spaces, Business Space displays the last open page.
- **Editing pages**: You can edit a page in it to add or remove widgets or change where the widgets are displayed in the page.
- **Deleting pages**: If you own a page, you can delete it from its business space. However, you cannot restore deleted pages. If you think you might need the page, export it. The page can then be restored by importing it.

IBM Training

Pages (2 of 2)

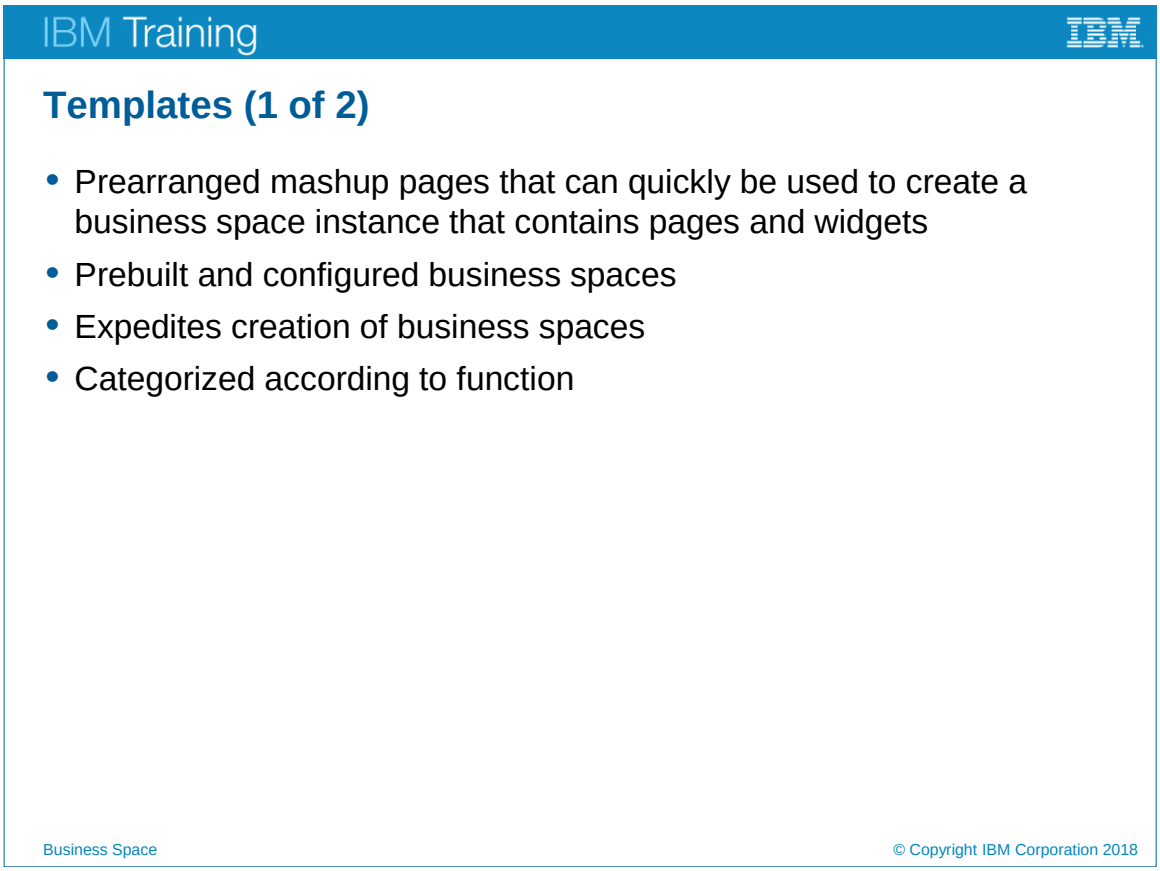
- Layout of widgets, their configuration, and an event flow between widgets
- Pages are also known as a mashup



Business Space

To organize the widgets on a business space page in a different pattern, or to use the screen area more efficiently, you can change the layout of the page.

A page acts as a container for one or more widgets. If more than one widget is placed on a page, it becomes a mashup. By definition, a mashup combines data from more than one source into a single integrated tool. This example shows six widgets that are placed on the page that is named My Work.



Templates

You can configure the business space more easily by using the preconfigured templates that come with a business space. You can access the templates when you create a business space. In addition to the provided templates, you can also create customized templates from business spaces. If you create a business space that you want to save as a template to share with other users, a superuser can convert the business space into a template.

The template can then be shared with other users. If any of these users change any of the pages in the customized template, then the changes are forwarded to other users who share the template. However, because future updates to the business space can overwrite this type of change, the superuser must create a copy of the original template before doing any modifications.

IBM Training

Templates (2 of 2)

- Product templates have widgets from a single product
  - The Create Space window lists only the product templates for the products that are installed
- Cross-product templates have widgets from more than one product
  - The Create Space window lists all of the cross-product templates
  - If a product is not installed, its templates are not available for selection when you are creating a business space
  - Templates must be in the same profile in WebSphere Application Server

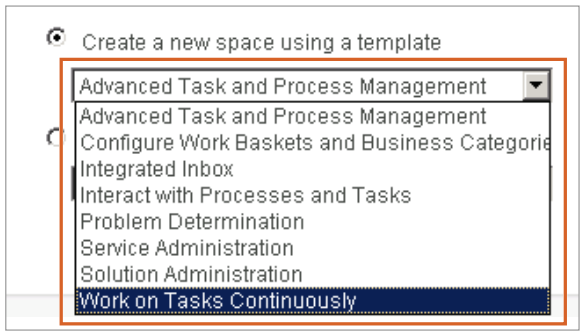
Business Space

Cross-product templates are templates that contain widgets from more than one product within the IBM BPM portfolio. Before users can create spaces that are based on these templates, the administrator must import them and make them available.

IBM Training

Templates available in IBM Process Manager

- Templates for managing processes and tasks:
  - Advanced Management of Tasks and Processes template
  - Configure Work Baskets and Business Categories template
  - Integrated Inbox template
  - Interact with the Processes and Tasks template
  - Work on Tasks Continuously template



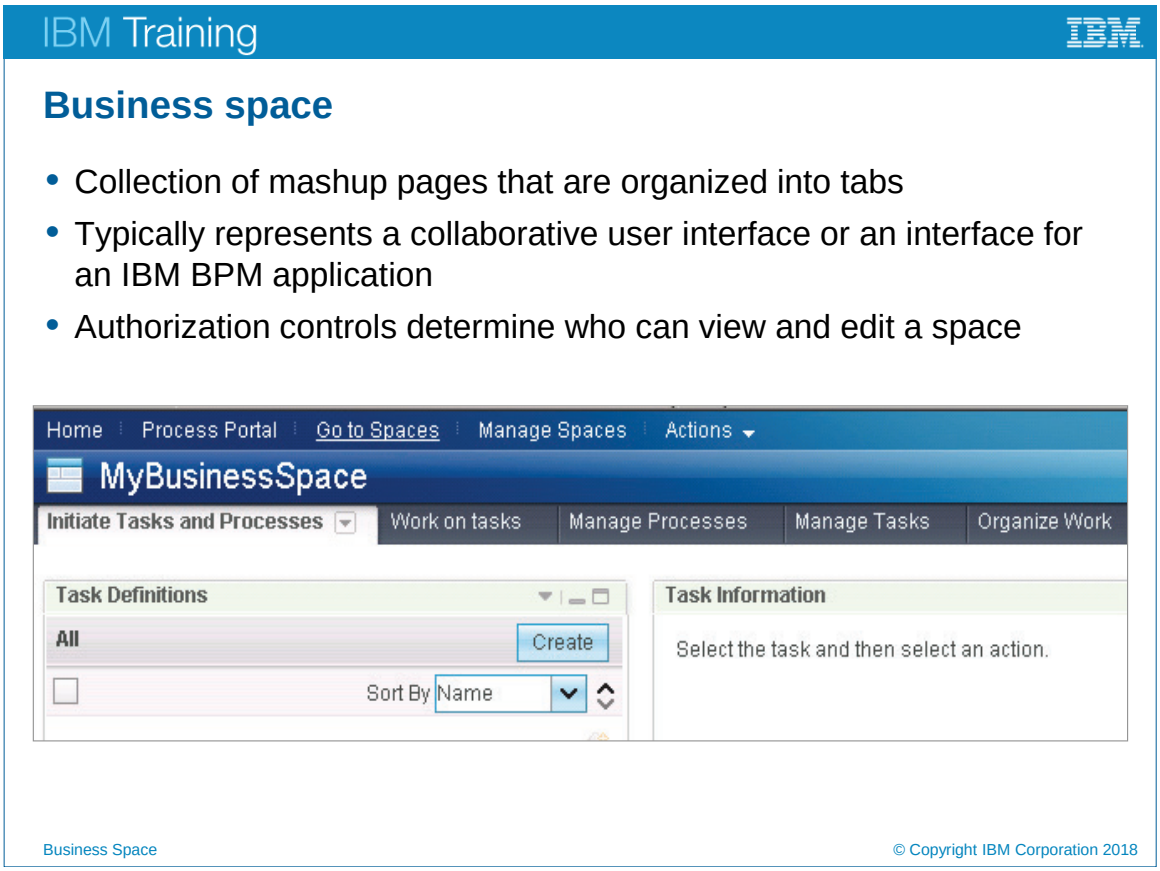
Business Space

Templates available in IBM Process Manager

Several templates are provided for you to create business spaces to manage processes and tasks. These templates support specific usage patterns for working with and managing tasks and processes.

- **Advanced Management of Tasks and Processes template**: Use the Advanced Management of Tasks and Processes template to create a space for business users and team leads who work on tasks and collaborate with other people. You can also use this template to organize your own task lists, check the status of processes you are involved with or responsible for, and check tasks that are related to these processes.
- **Configure Work Baskets and Business Categories template**: Use the Configure Work Baskets and Business Categories template to create a space for users who are responsible for the configuration and management of work baskets and business categories within the organization.
- **Integrated Inbox template**: Use the Integrated Inbox template to create a space for people who must work with both IBM Business Process Manager tasks and IBM Case Manager work items. Users of this space can work on tasks and work items from their inbox.

- **Interact with Processes and Tasks template**: Use the Interact with Processes and Tasks template to create a space for people who use a list of tasks – for example, an inbox – as the basis for their work. Users of this space can also create processes and services, and find detailed status information for specific process instances.
- **Work on Tasks Continuously template**: Use the Work on Tasks Continuously template when you must create a space for users who continuously work on tasks for a task queue. When users complete tasks, they get the next available task in the queue assigned to them automatically.



Business space

A business space is a collection of related web content that provides you with insight into your business and the capability to react to changes in it.

In Business Space, you can have many business spaces with each one having a different purpose. For example, a business space with widgets from IBM Business Monitor might be used to monitor key performance indicators in your business and widgets from IBM Process Manager to manage the tasks that people do. Business Space can display the contents of one business space at a time. This space is the open space.

The open space consists of a banner area and a page area. The top of the banner consists of menus and links. The menus contain options that affect the open business space or options for browsing to another space. The links in the top banner go to the help or log you out of Business Space. Below the menus, the banner area displays the name of the open space and tabs for the pages in the space. The page area is below the banner and displays an open page. The tab for the page is highlighted.

If you are the owner of the open page or an editor, an **Edit Page** button is also available. When you click **Edit Page**, the editing toolbar opens. The toolbar contains the palette of widgets and several buttons. By using these buttons, you can see the hidden widgets on the page and save changes.

The toolbar also contains two fields that you can use to filter widgets from the palette. One field removes all widgets that do not belong to a selected category, while the other removes all widgets that do not contain the characters that you specify in their name or description.

The page area is where Business Space displays the visible widgets on the page.

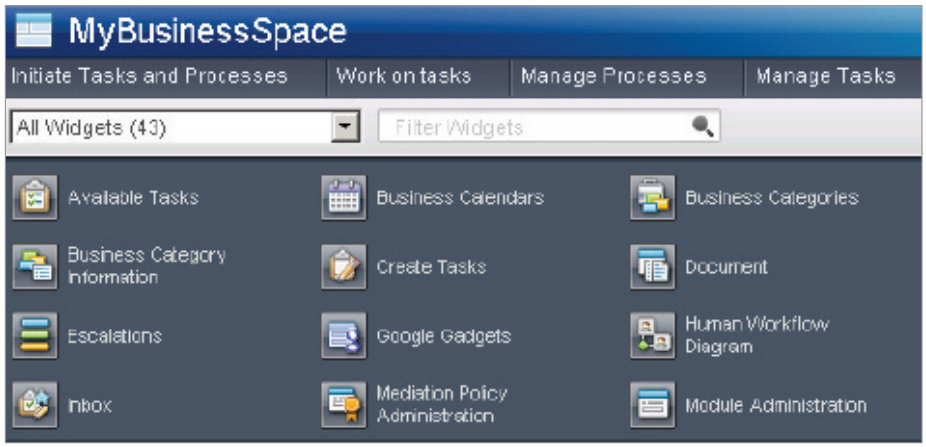
You can create a business space in any of the following ways:

- Using the widget palette to define the tasks
- Using a preconfigured template
- Importing an existing business space

IBM Training

Widget

- Configurable piece of a graphical user interface function, provided as an embeddable component
- Multiple instances can be on the same mashup page, each with a different configuration



Business Space

Widget

A business space comprises one or more pages. Each page contains one or more widgets.

Widgets are the pluggable user interface components that you use to define the functions of your business spaces.

Typically, a single widget has limited or specific capabilities. However, you can combine widgets to interact with each other to do related tasks. Combined widgets are called a mashup. For example, you can have a widget that displays news items from an RSS feed and another widget that you can use to create tasks that you assign to someone. If you combine the two widgets onto one page, you create a mashup. In this mashup, you can see a news event and react to it by assigning someone to investigate whether the event might affect your business. Some widgets communicate with other widgets so that the events in one widget affect the contents of a different widget.

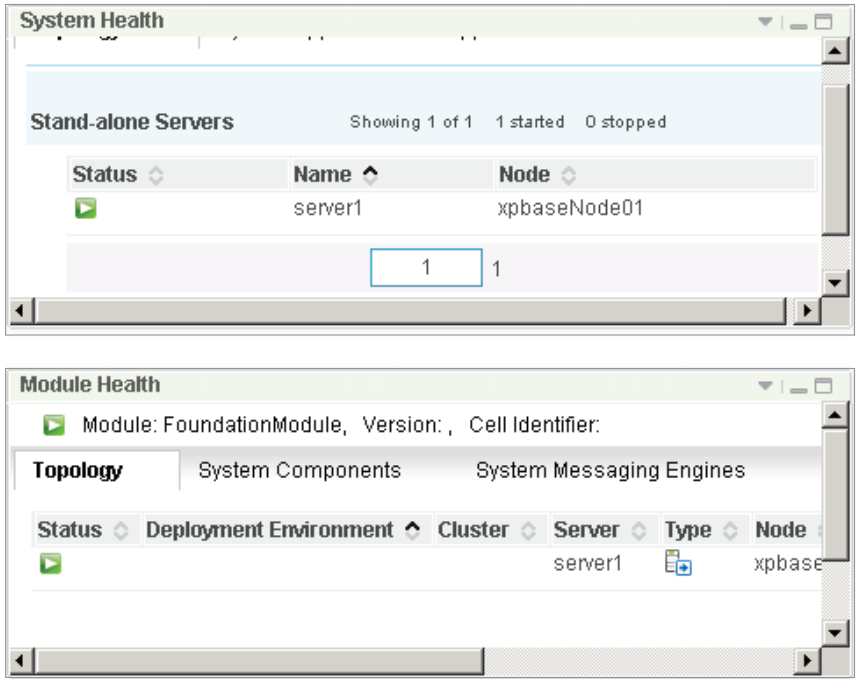
You can minimize, maximize, and drag widgets while laying out a page. In addition, each widget has a menu that contains actions that you can take on the widget and a Help option, which links to information about that widget. In addition, you can convert a widget into a hidden widget. A hidden widget is fully functional but is not displayed on a page. Typically, you hide widgets such as the Script Adapter widget, which, instead of displaying server data, transforms business data so that another widget can use it.

Other than not being displayed on a page, a hidden widget is like a visible widget. That is, you can change its settings and wire it to other widgets.

Business Space provides a widget palette that contains categories of widgets that you can use to configure the pages in your business spaces. You can use the categories to filter out the widgets that are not in the category that you select. You can see all of the available widgets by selecting **All Widgets** from the list of categories.

IBM Training

Problem determination widgets



Business Space

Problem determination widgets

Administration widgets offer a way to manage and monitor the individual components of your overall business process management solution, including modules and services.

The administration widgets are grouped on pages in the following templates:

- The Solution Administration template contains the widgets that you need for administering the modules in your solution.
- The Service Administration template contains the widgets that you need for monitoring and working with the services that are used in your solution.
- The Problem Determination template contains the widgets that you need for monitoring the health of your modules and of your overall system.

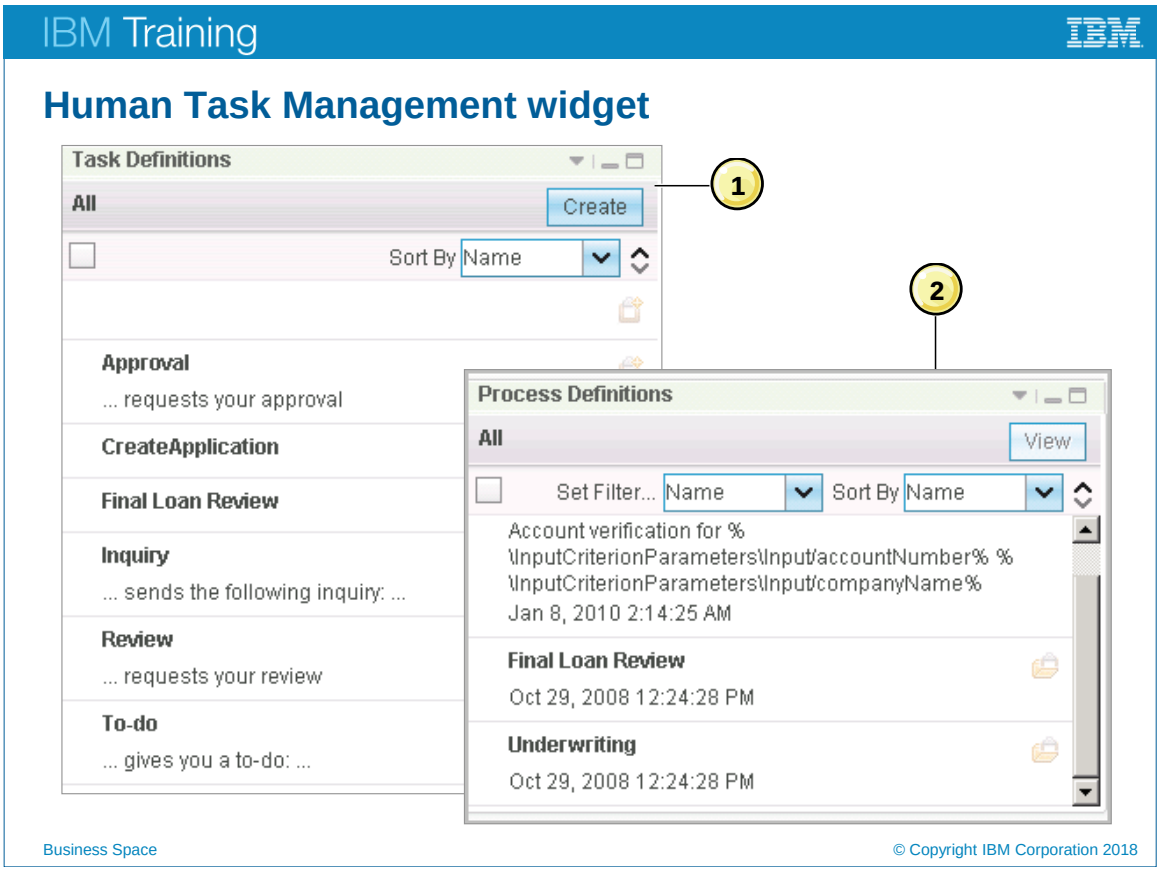
These widgets are delivered as part of Business Space and function in the same way as other widgets. You can minimize, maximize, and drag widgets while laying out a page. In addition, each widget has a menu that contains actions that you can take on the widget.

The System Health widget and the Module Health widget are commonly used problem determination widgets.

- **System Health** widget: Use the System Health widget to view a snapshot of the overall system health of your business solution. This widget provides a single place from which you can quickly assess the status of application servers, nodes, clusters, deployment environments, messaging engines and their queues, databases, system applications, and failed events.
- **Module Health** widget: Use the Module Health widget to evaluate the health of your module and identify potential problems. The widget provides a central place for health information about module topology, system components, system messaging engines, queues, data sources, and failed events.

Module Health interacts with the Module Browser widget. When you select a module from Module Browser, the Module Health widget refreshes to display health information for the selected module. The Module Health widget displays an overall status for the module. It also groups more status and health information for major areas of your module, organizing it into tabs. If a resource encounters a problem (for example, a queue that reaches capacity, or a node agent that is not running), a warning icon is displayed at the top of each affected tab.

Note: If administrative security is enabled for Business Space, you must also enable application security for Module Browser to work properly.

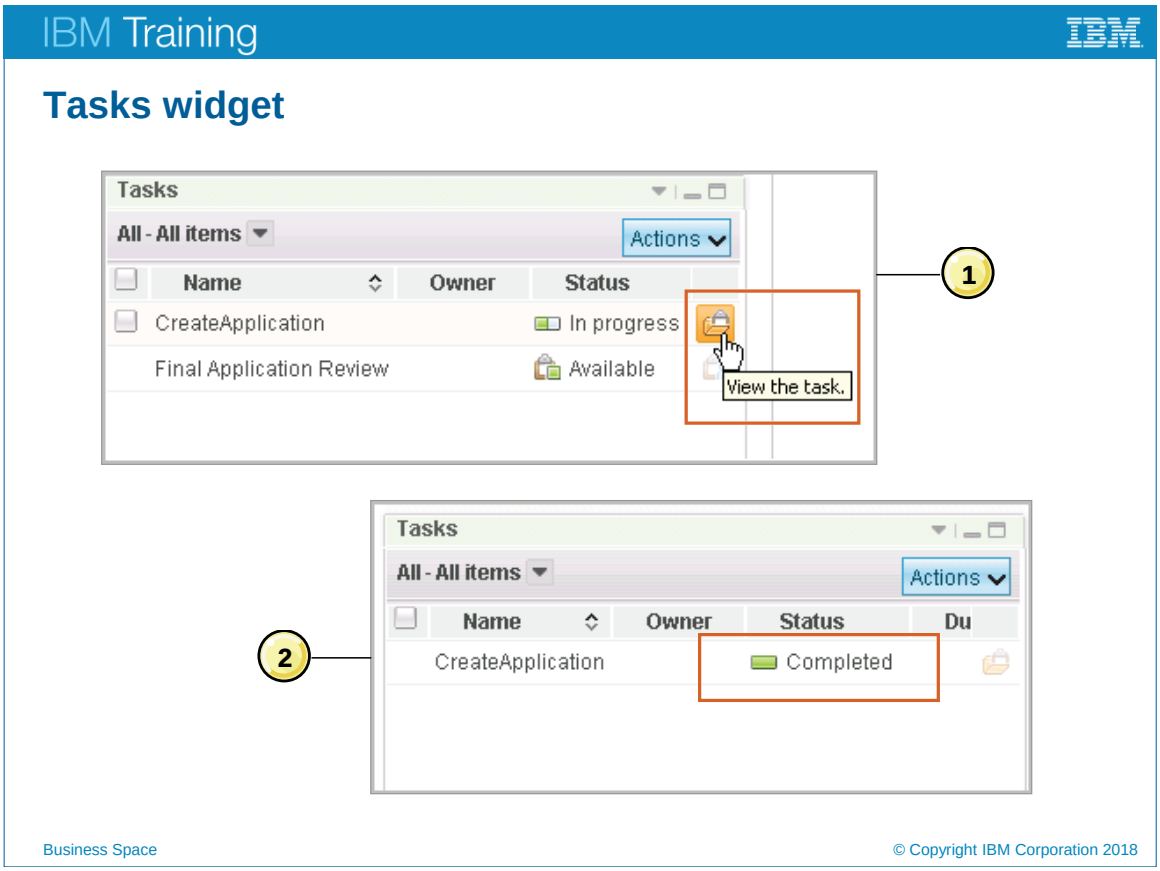


Human Task Management widget

The Human Task Management widgets provide business users and managers with capabilities to check and explore processes and human tasks. They can easily be combined with other widgets to compose powerful business spaces.

These widgets are delivered as part of Business Space, and they function in the same way as other widgets. You can minimize, maximize, and drag widgets while laying out a page. In addition, each widget has a menu that contains actions that you can take on the widget. The widget menu also has a help icon, which links to information about that widget.

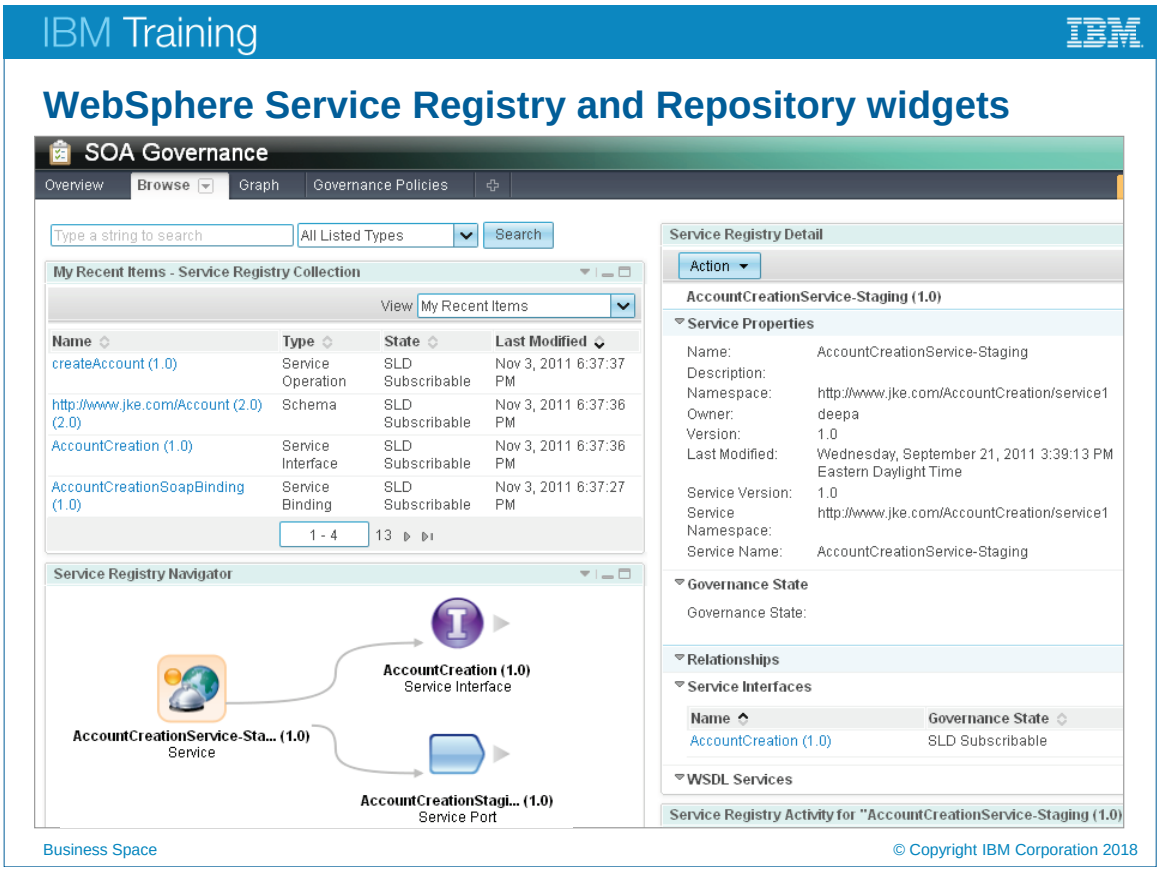
1. Task Definitions widget: Task-related widgets allow business users to work with and manage tasks and escalations. The Task Definitions widget belongs to the group of task-related widgets. Use the Task Definitions widget as the starting point for creating tasks and working on tasks that are associated with specific task definitions.
2. Process Definitions widget: Process-related widgets allow business users to work with and manage processes. The Process Definitions widget belongs to the group of process-related widgets. Use the Process Definitions widget as the starting point for working on processes that are associated with specific process definitions.



Tasks widget

Use the Tasks widget to work with different types of tasks; for example, tasks that you own, or tasks that are available for you to work on.

1. You can click the task icon to view the task so you can work on it. Depending on the status, the status might be “In progress” or “Available” when you are ready to work on it.
2. When you are done with the task, the status changes to “Completed.”

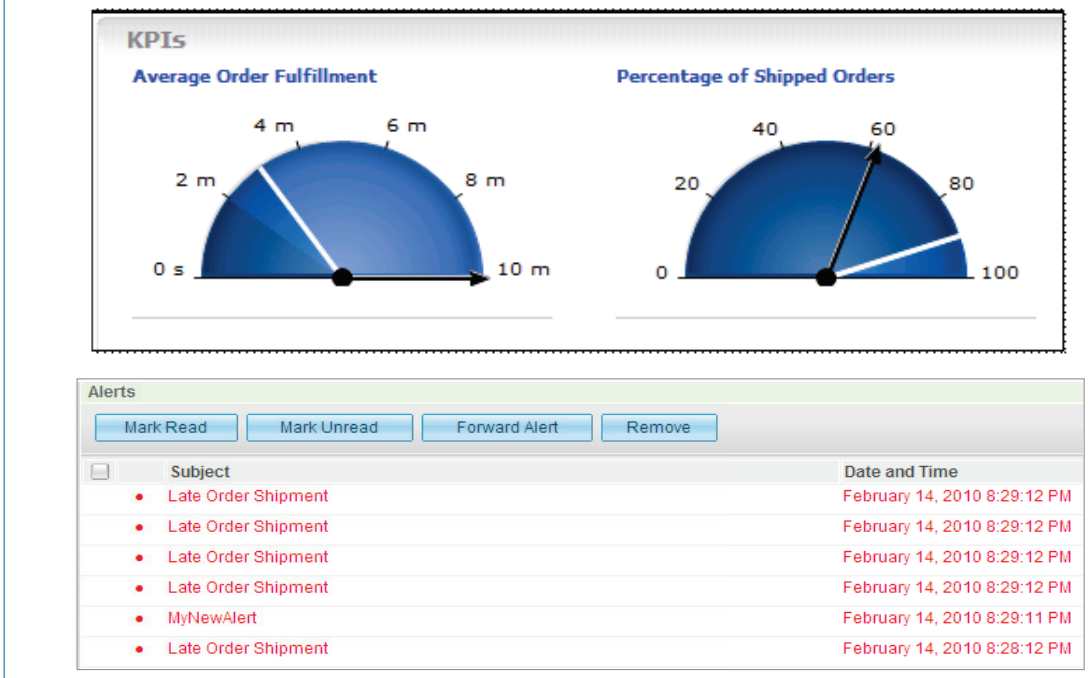


WebSphere Service Registry and Repository widgets

The slide depicts several widgets in WebSphere Service Registry and Repository. Before you can create such a space with widgets, WebSphere Service Registry and Repository must be installed and configured in the environment.

IBM Training

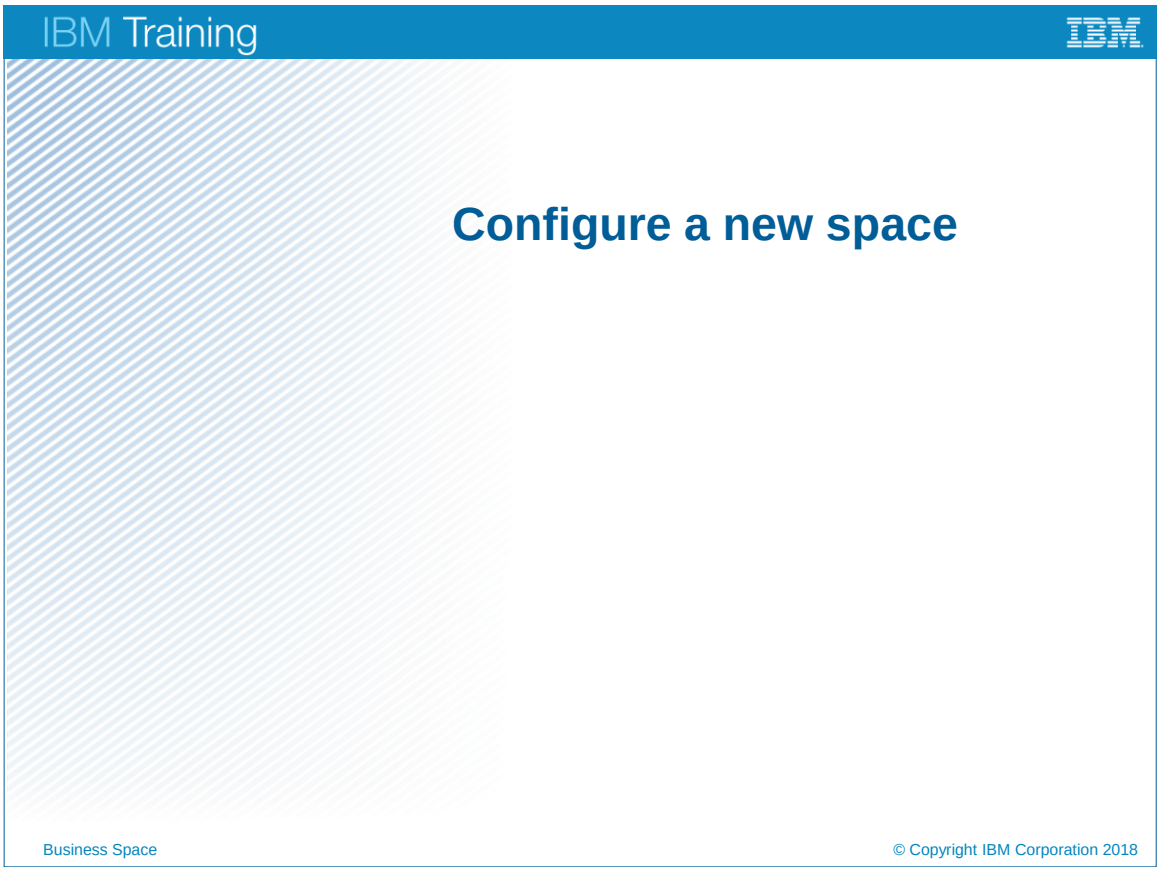
IBM Business Monitor widgets



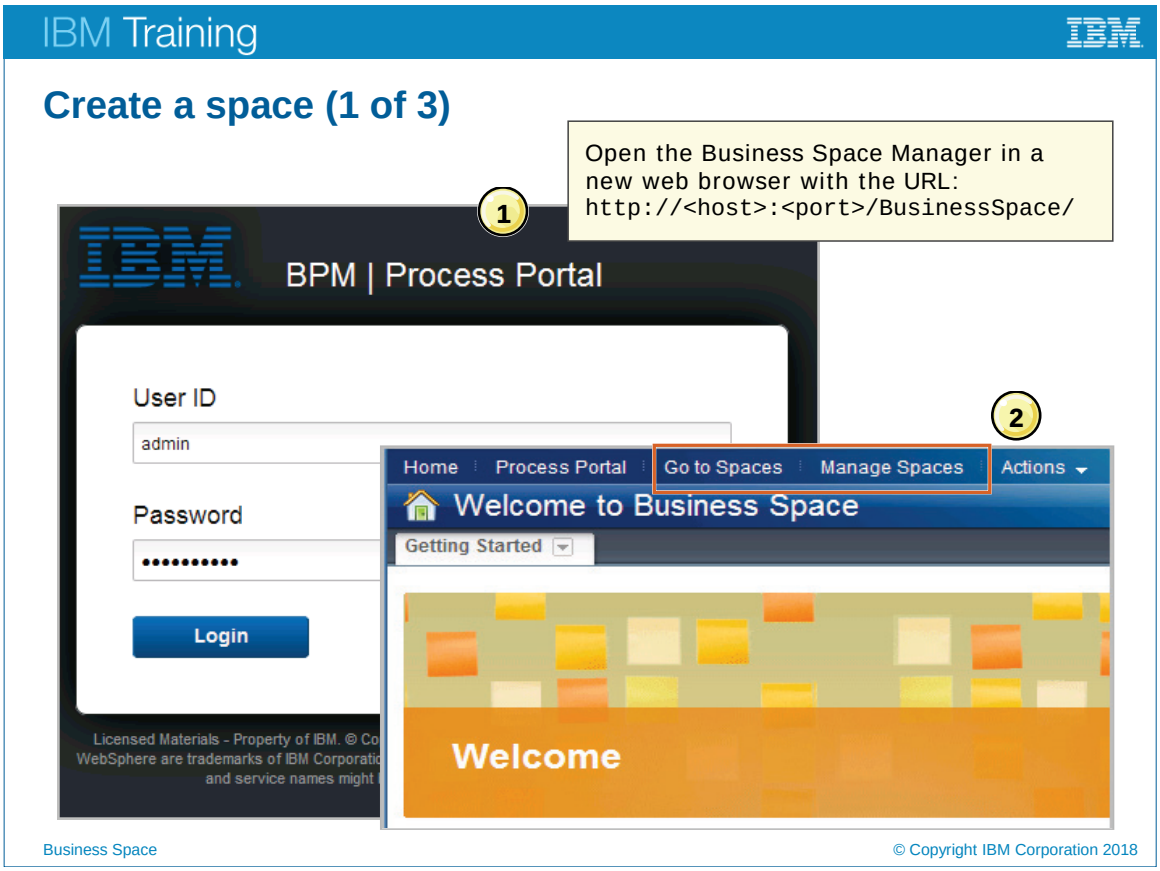
Business Space

IBM Business Monitor widgets

The slide depicts several widgets in IBM Business Monitor. Before you can create such a space with widgets, you must install and configure IBM Business Monitor in the environment.



Configure a new space



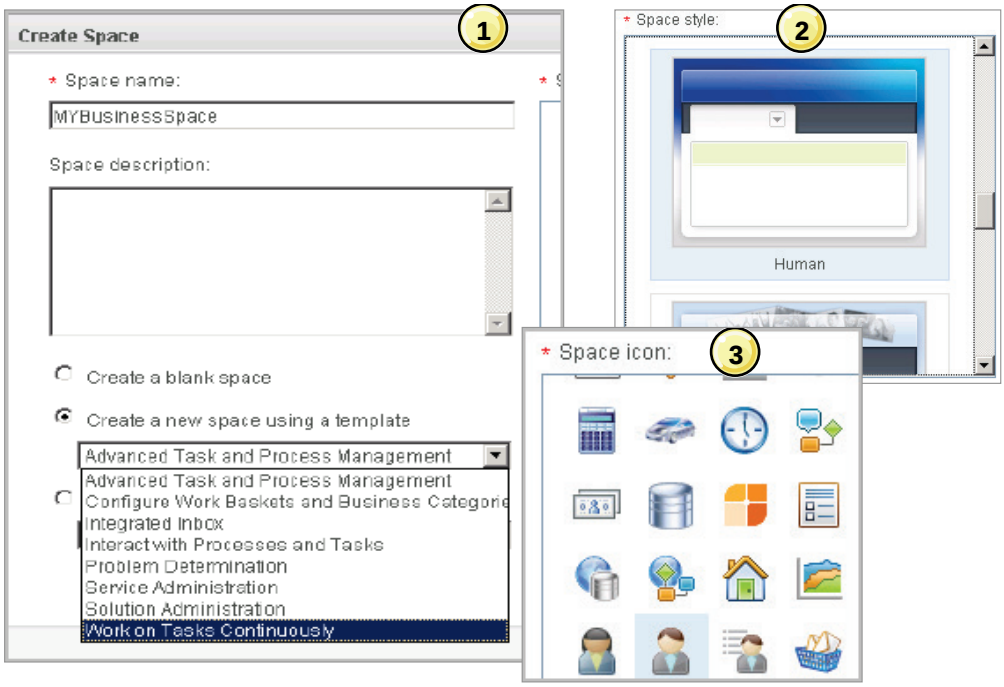
Create a space

To start Business Space, you open a browser and enter `http://<host>:<port>/BusinessSpace/` in the URL field.

1. The login page for Business Space and Process Portal is the same.
2. After logging in, you can click **Go to Spaces** to go to a specific space or click **Manage Spaces** to create a new space.

IBM Training

Create a space (2 of 3)



Business Space

A business space is a collection of related web content that provides you with insight into your business and the capability to react to changes in it. To contain a collection of related pages, you can create a business space.

1. In the Create Space window, you must provide a unique space name that describes your business space. When you create a space, the title and description that you provide serve as default values in all languages. You can then switch languages and replace those default values with the appropriate translations.

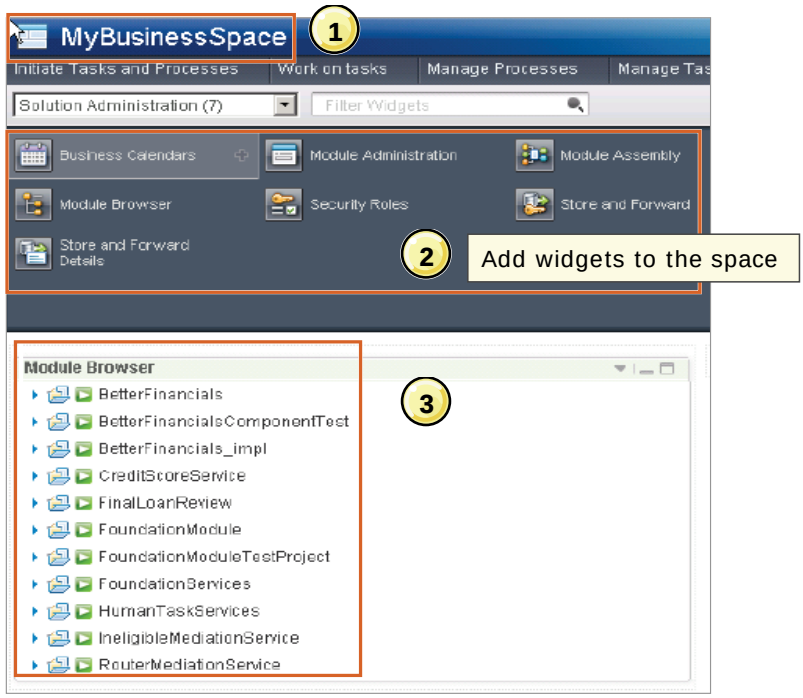
   Select how you want to create the space:

   - Create a blank space, which creates an empty space.
   - Create a space by using a template, which creates a space that contains the pages and widgets that the template defines. Each template supports one or more scenarios by providing the widgets that you need for the scenarios and organizing the widgets on the pages.
   - Duplicate an existing space, which creates a space that contains the same pages and widgets as an existing space.

2. Select a style to set the look for the pages in the new business space, and select an icon to represent the space. In Business Space running on IBM WebSphere Portal Server, the business space automatically has the default Business Space style and icon applied. You can select a different style only when you are editing a page, and you can select a different icon only when editing the space settings.
3. A list of icons is available for selection for your business space. This icon is displayed along with the title on the banner of the business space page.

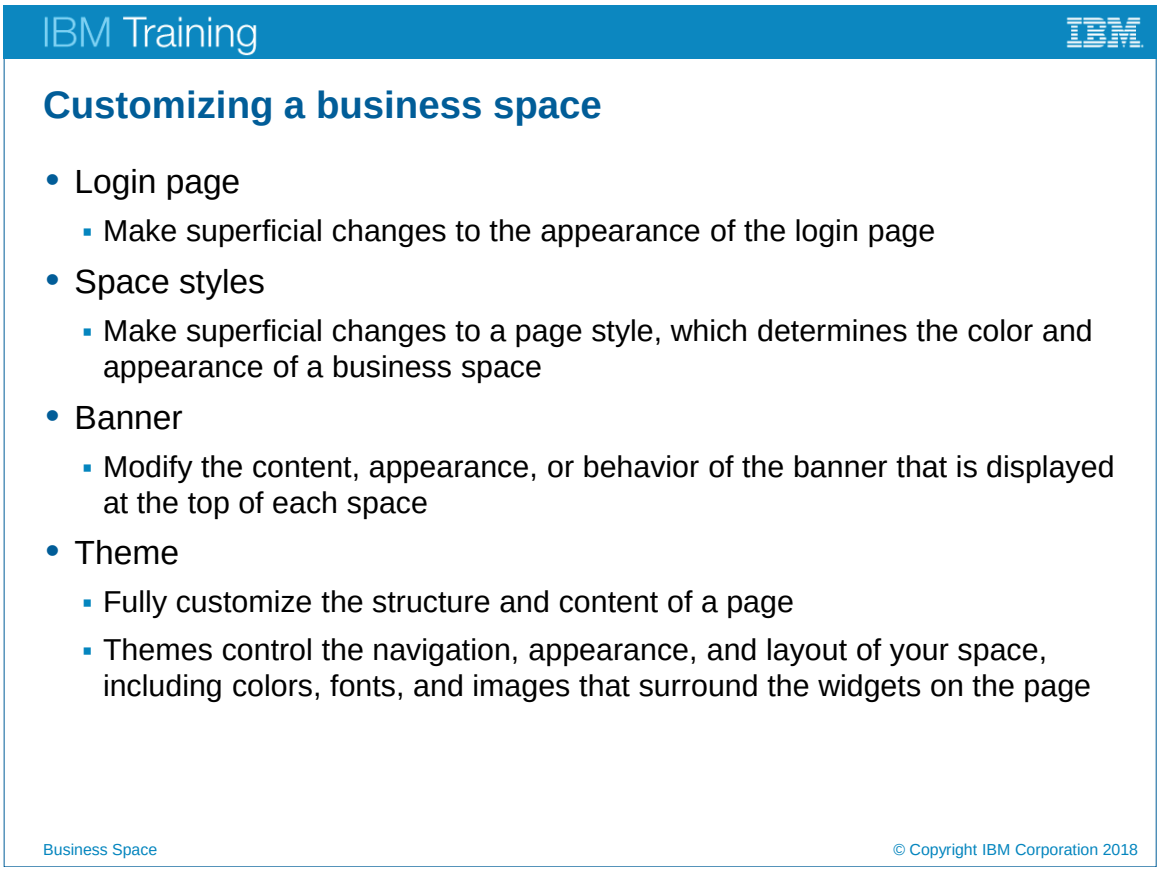
IBM Training

Create a space (3 of 3)



Business Space

1. The business space title and icon are displayed in the banner area.
2. Select from the available widgets in the widget section and add to the page. You must click the **Edit Page** link before you can select and add the widgets.
3. The widget is displayed in a page. You can also add multiple related widgets on a single page.



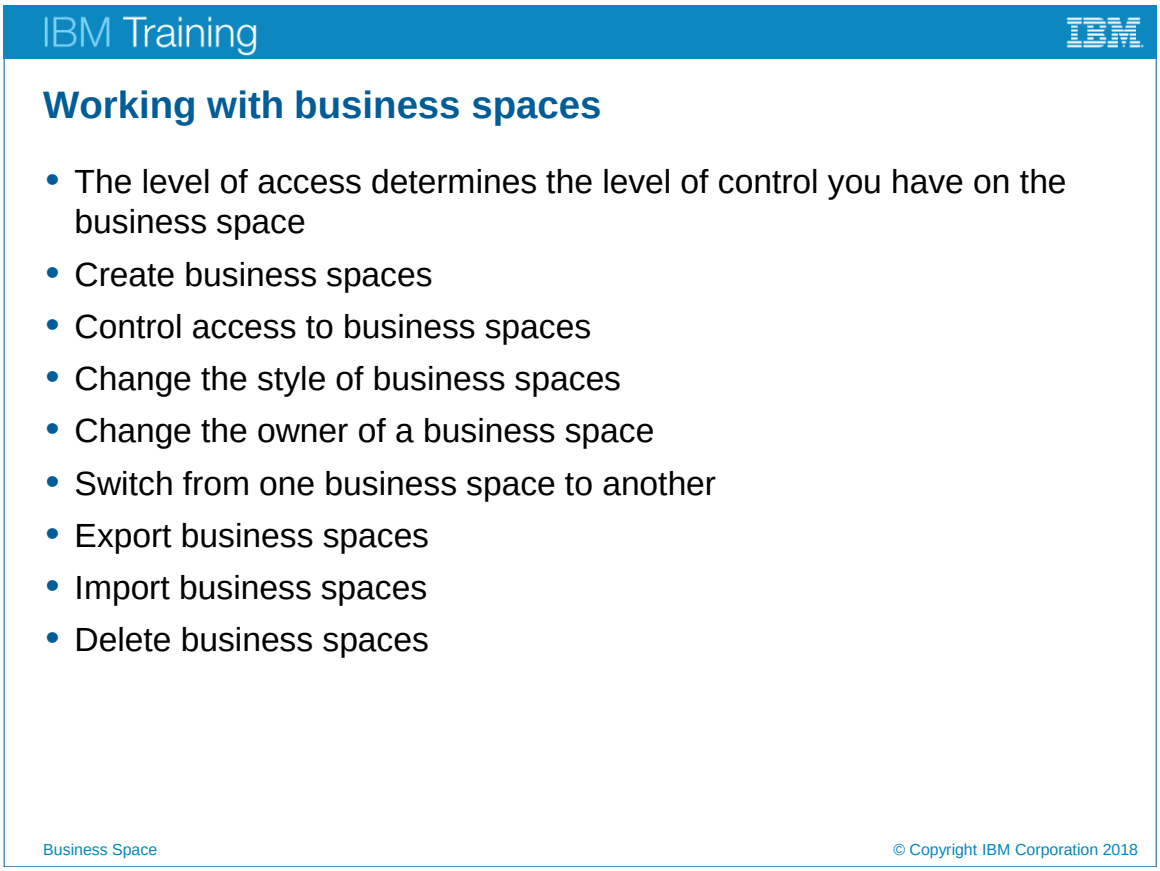
Customizing a business space

You can customize the behavior and the appearance of your business space in various ways, from small changes like modifying default link text to large-scale changes like customizing the entire theme.

Business Space supports the customization of these areas:

- **Login page**: You can make superficial changes to the appearance of the login page.
- **Space styles**: You can make superficial changes to a page style, which determines the color and appearance of a business space.
- **Banner**: You can modify the content, appearance, or behavior of the banner that is displayed at the top of each space.
- **Theme**: You can fully customize the structure and content of a page. Themes control the navigation, appearance, and layout of your space, including colors, fonts, and images that surround the widgets on the page.

Web-based Distributed Authoring and Versioning (WebDAV) is used to store and deploy the artifacts that are used for the login page, styles, banners, and themes. If you customize any of these artifacts, deploy them using WebDAV. WebDAV is a set of extensions to Hypertext Transfer Protocol (HTTP) which allows users to cooperatively edit and manage files on Remote Web servers. Most operating systems provide built-in support for WebDAV.



Working with business spaces

Depending on your level of access, you can create business spaces, modify existing ones, and delete obsolete spaces.

- **Creating business spaces**: To contain a collection of related pages, you can create a business space.
- **Controlling access to business spaces**: If you are the owner of the business space, you can share it so that others can access it. As part of the process of sharing the space, you can set who is able to view and edit the business space.
- **Changing the style of business spaces**: If you can edit a business space, you can change its style so that it has a different color and appearance. The method that you use depends on whether you are using Business Space running on WebSphere Application Server or Business Space running on WebSphere Portal Server.
- **Changing the owner of a business space**: If you own a business space, you can transfer ownership of that business space to another person.
- **Switching from one business space to another**: When you are finished with a business space, you can switch to another business space. When you switch to another space, the first space closes automatically.

- **Exporting business spaces**: If you can edit a business space, you can export it. Exporting a space is useful if you want to give someone a copy of your space without giving them access to it or if you want to re-create the space later.
- **Importing business spaces**: You can import a business space and the pages that it contains. Importing is a way of creating a business space that is already populated with pages and configured widgets.
- **Deleting business spaces**: If you own a business space, you can delete it so that it and the pages it contains are no longer available.

IBM Training

Unit summary

- Describe the purpose and business value of Business Space
- Describe the types of widgets that are available in Business Space
- Describe the types of templates that are available in Business Space
- Explain how to create and configure a new space in Business Space

Unit summary

IBM Training

Checkpoint questions

1. True or False: A page must be created for every widget that is added to it.

2. If only the templates specific to IBM Process Manager are available in Business Space, what can be done to add more templates?

3. True or False: Multiple pages can be added to a business space.

Checkpoint questions

IBM Training

Checkpoint answers

1. False. You can add multiple widgets with different capabilities on the same page. You can also combine widgets to interact with each other to work on related tasks. Combined widgets are called a mashup.

2. Cross-product templates are templates that contain widgets from multiple products. The administrator must first import the templates and make them available before users can create spaces that are based on these templates.

3. True. Multiple pages can be added. However, if you have too many pages in a space, it affects performance. The maximum is 10 pages. Try to limit the pages in a space to the ones that are necessary for the space.

Checkpoint answers

IBM Training

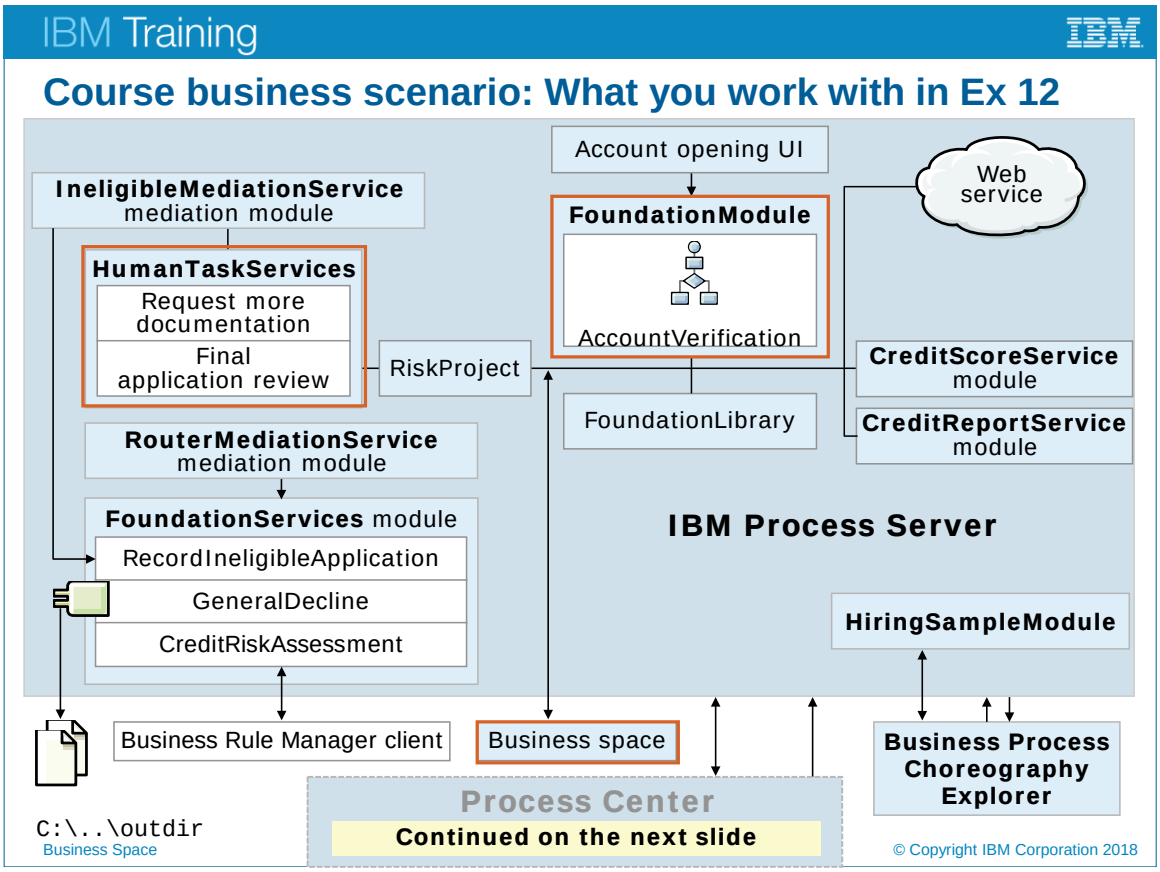
Exercise 12: Exploring Business Space

After completing this exercise, you should be able to:

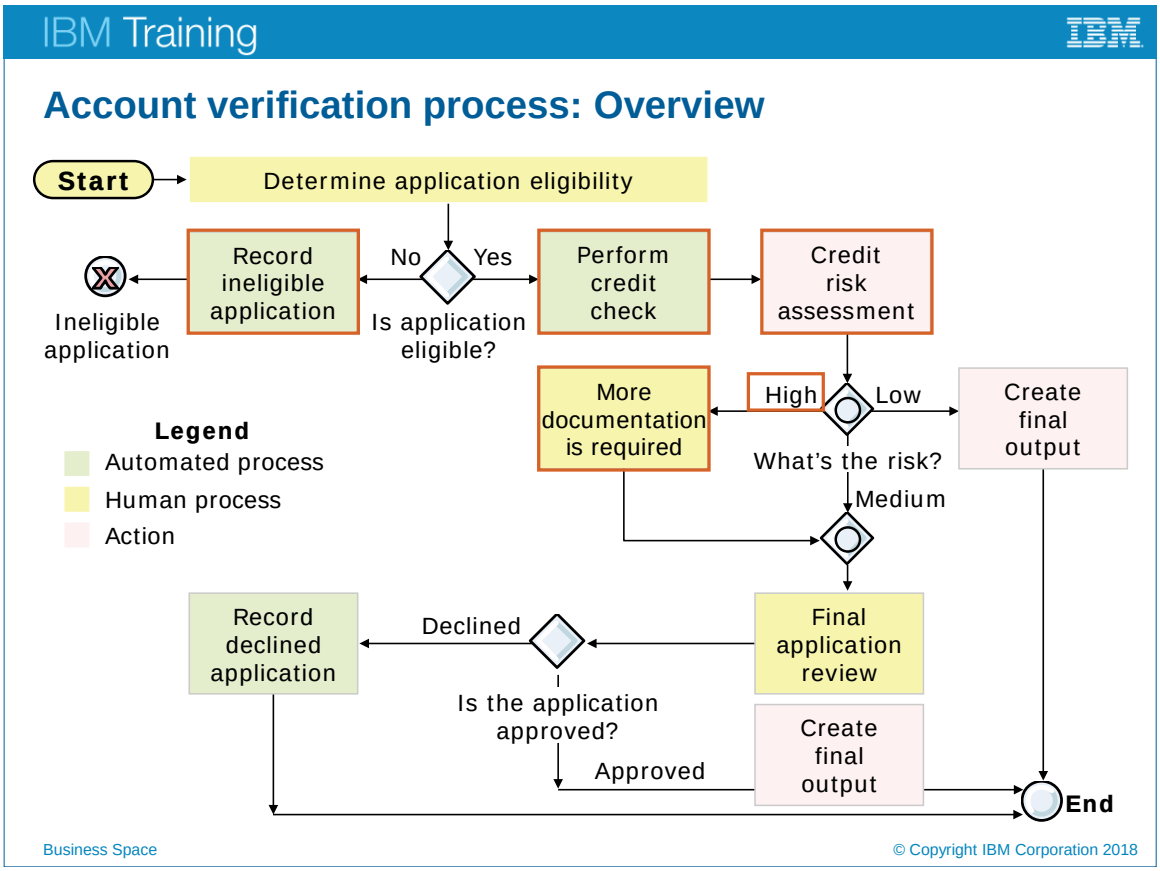
- Create a space by using the Business Space client
- Use the Business Space client to work with human tasks
- Create a page and display content by using widgets and Business Space templates

Business Space

Exercise 12: Exploring Business Space



Course business scenario: What you work with in Ex 12



Account verification process: Overview

Business Space is a browser-based graphical user interface that you use to view and interact with content from various products in the business process management portfolio. Business Space provides a single web-based point of access for the content, and you use Business Space to combine the content in useful and interesting ways. These combinations give you insight into your business and the capability to react to changes in it.

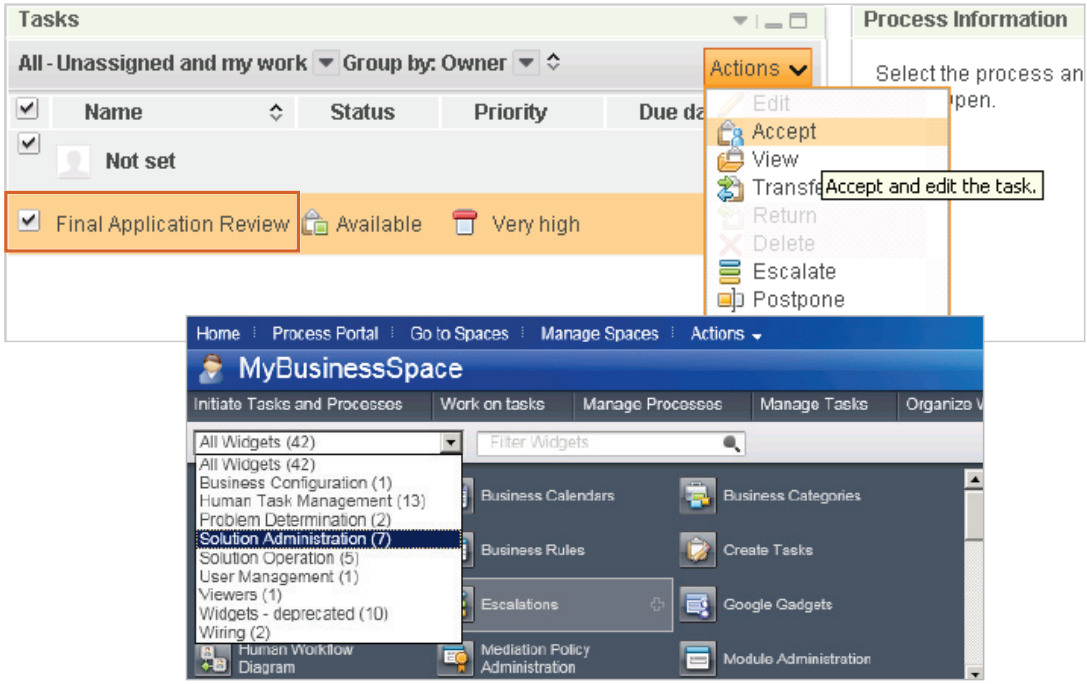
In this exercise, the entire “create account” solution is assembled, and you can run end-to-end tests of the applications. Different possible paths are available through the AccountVerification business process. You run a test to verify a test case to work with the Business Space client.

When you use company name ACME to test the applications, the eligibleApplication attribute is set to true and the creditRisk evaluates to MED. The creditScore returned is 6. When you use companyName ACME to submit an application, the application flows through these activities: **Account Verification Receive > Determine Application Eligibility > Map to Credit Check > Credit Check Service > Map Credit Checking Result > Credit Risk Assessment > Assign Variable > While More Documents Required > Request More Documentation**.

A user interface for Request More Documentation is used to change the comment field from None to Complete. After leaving the While More Documents Required loop, the application flows through **Merge Assign > Create Output > Final Application Review**. A user interface for Final Application Review is used to update the applicationDecision field. If applicationDecision is true, the application flows to **Create Output > Account Verification Reply**. If applicationDecision is false, the application flows through **Generate Decline > Record Declined Application > Account Verification Reply**.

IBM Training

Create business space in Exercise 12



Business Space

Create business space in Exercise 12

Exercise 12: Exploring Business Space

Purpose:
In this exercise, you explore the capabilities of Business Space by using templates, pages, and widgets.

Business Space is an integrated user experience for business users across the IBM Business Process Manager portfolio. Business Space provides a customizable and collaborative environment for you to monitor, review, and administer common business processes, such as human task flows, modeling, and performance indicators.

Business Space is a browser-based graphical user interface that you use to view and interact with content from various products in the business process management portfolio. Business Space not only provides a single web-based point of access for the content, you use Business Space to combine the content in useful and interesting ways. These combinations give you insight into your business and the capability to react to changes in it.

Requirements

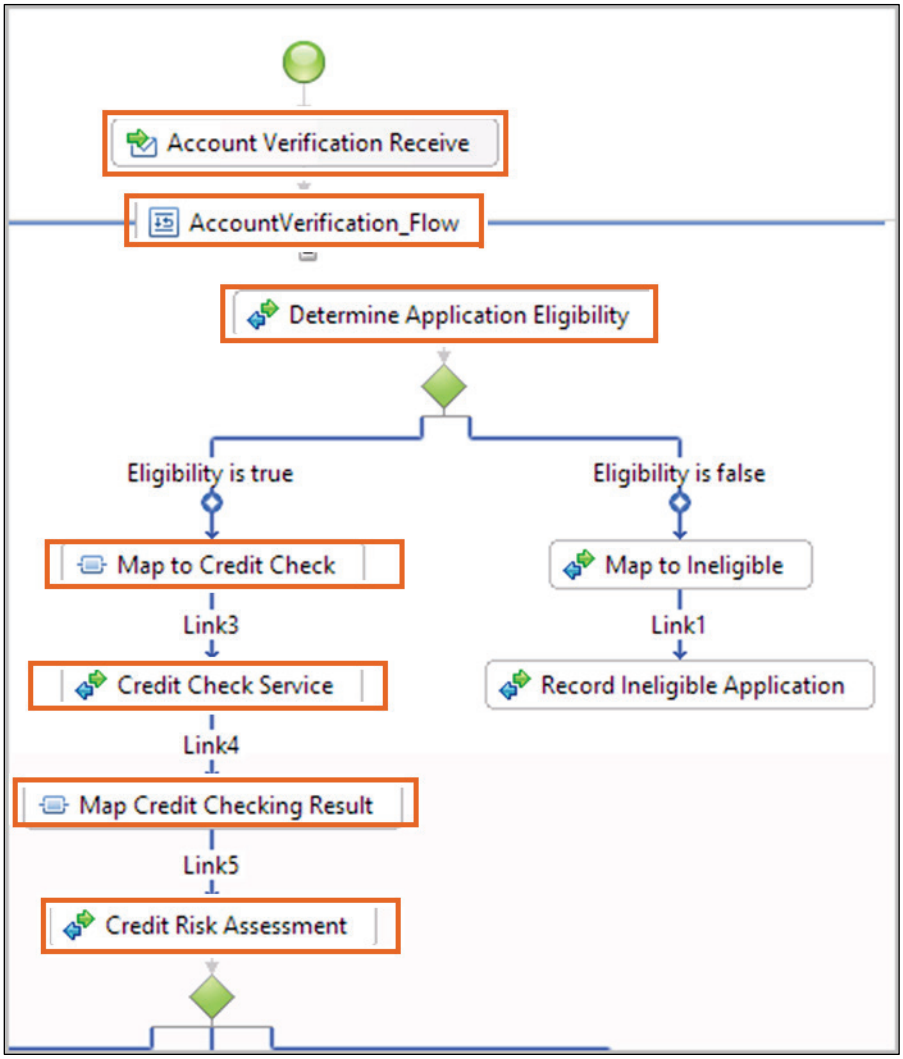
Completing the exercises for this course requires a lab environment. This environment includes the exercise support files, IBM Process Designer, IBM Process Center, and IBM Integration Designer test environment.

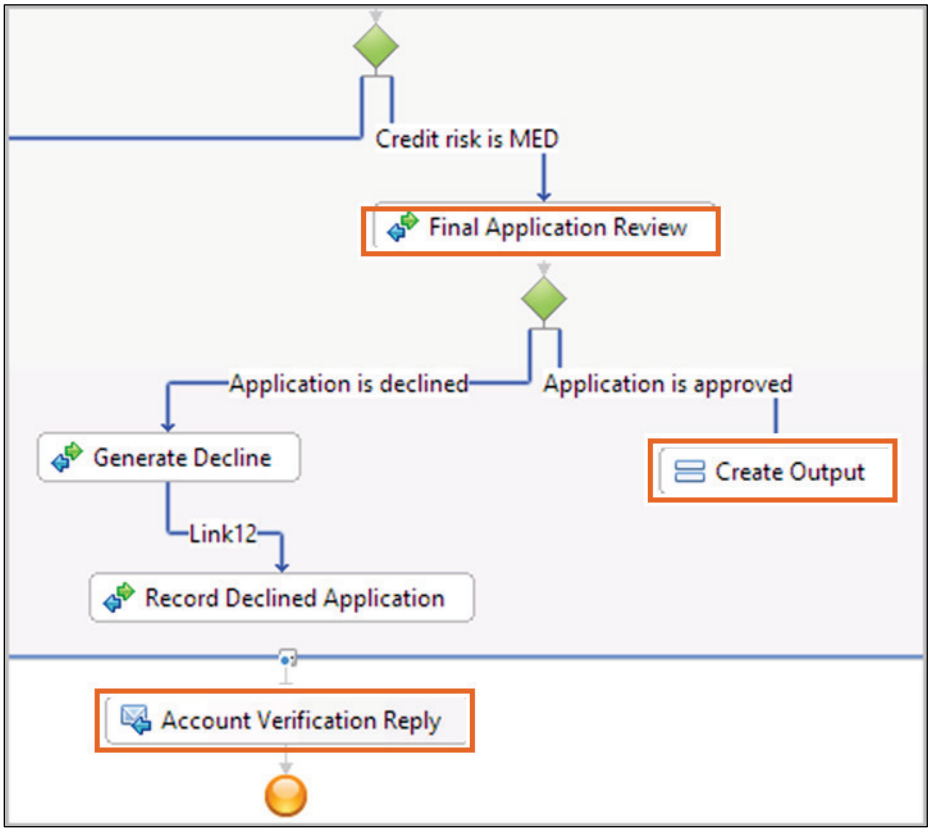
Part 1. Use the Business Space clients to work with human tasks.

In the workspace for this exercise, the entire “create account” solution is assembled, and you can run end-to-end tests of the applications. You can use different possible paths through the AccountVerification business process. You run a test to verify a test case to work with the Business Space client. You must familiarize yourself with the overall project to understand how data flows through the applications.

1. Use Business Space to test for eligible applications with MED credit risk

When you use company name ACME to test the applications, the eligibleApplication attribute is set to `true` and the creditRisk evaluates to MED. The creditScore returned is 6. When you use companyName ACME to submit an application, the application flows through these activities: **Account Verification Receive > Determine Application Eligibility > Map to Credit Check > Credit Check Service > Map Credit Checking Result > Credit Risk Assessment > Final Application Review > Create Output > Account Verification Reply**.



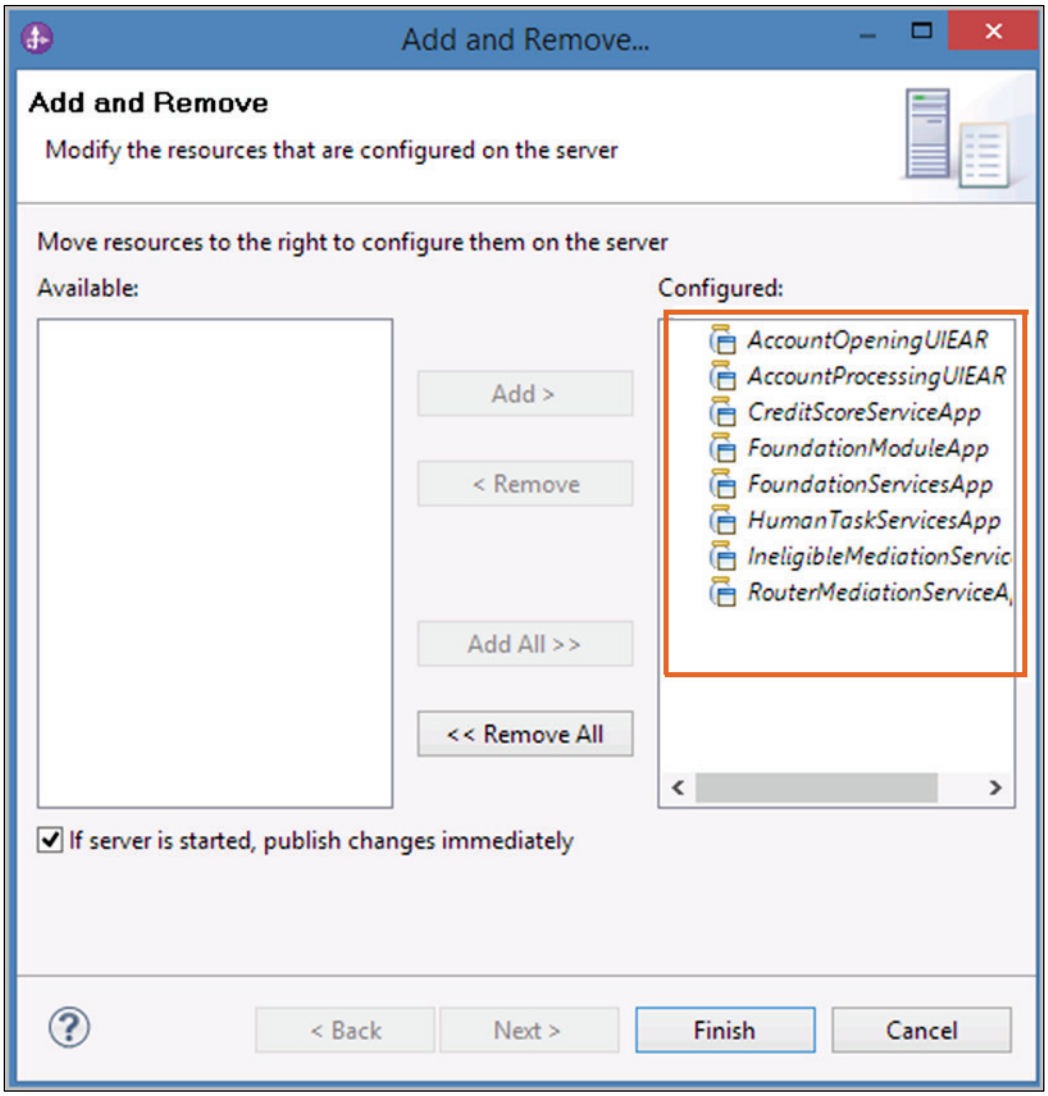


A user interface for Final Application Review is used to update the applicationDecision field. If applicationDecision is `true`, the application flows to **Create Output > Account Verification Reply**.

To test an eligible application with a `MED` credit risk:

1. On your desktop, open the **Exercise Shortcuts** folder.
2. Double-click the **Exercise 12** shortcut. Allow Integration Designer a few moments to build the workspace. You can view the workspace build status at the lower-right corner of Integration Designer. Wait until the status reaches100%, at which point the workspace is built, and the status progress bar disappears.
3. When the **Workspace Migration** window is displayed, clicked **Next**.
4. Click **Finish** to complete the migration.
5. Click **OK** in the **Migration Validation** window.
6. Wait for the workspace to build.
7. Close the **Getting Started** tab.
8. Double-click the **Start UTE Process Server** desktop shortcut to start the Process Server, if it is stopped

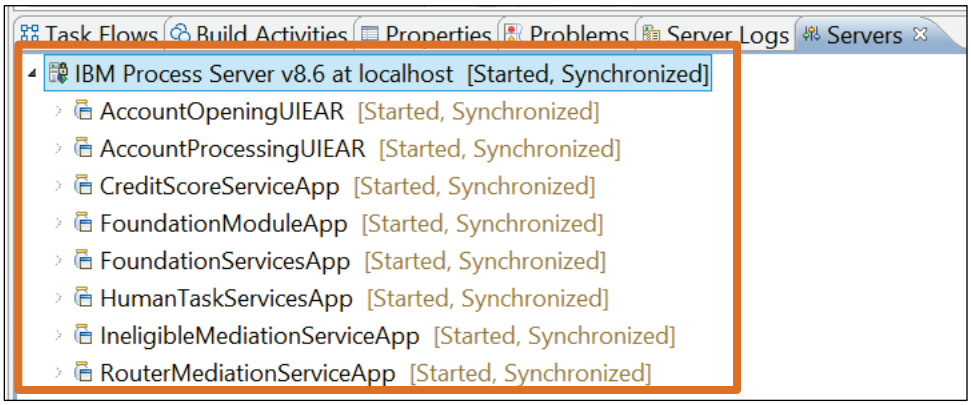
9. In the **Servers** view, right-click **IBM Process Server v8.6 at localhost** and click **Add and Remove**.
10. Click **Add All** to move each of the projects to the **Configured** list.



11. Click **Finish**.
12. Wait until the modules are published and started. It can take several minutes, depending on your system resources. To see the status of the modules, expand **IBM Process Server v8.6 at localhost** in the **Servers** view. Verify that the status of all the modules added is **Started**.
13. **Troubleshooting**: Occasionally, the status of the server and modules does not automatically refresh. It is possible that, while the server and modules are started, the status remains **Publishing**.

    Press F5 to refresh the view. Also, notice that often a module that is added has a **Stopped** status. To change its status, right-click the module, click **Restart**

from the menu, and republish the module if prompted. Repeat the step if necessary to ensure that the module starts.



Part 2. Creating a business space (your own workspace) in the Business Space user interface.

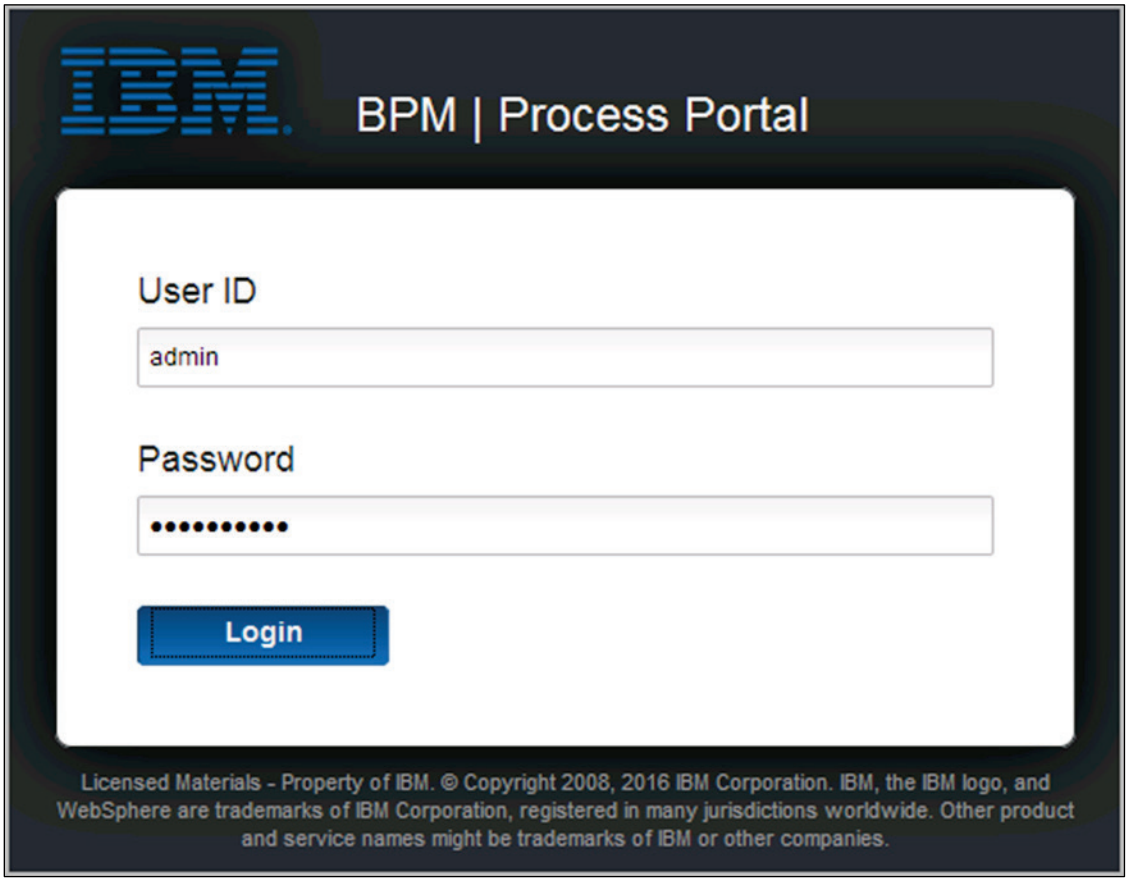
In Business Space, you can have many business spaces with each one having a different purpose. For example, a business space with widgets from IBM Business Monitor can be used to monitor key performance indicators in your business. Similarly, widgets from IBM Business Process Manager can be used to manage the tasks that people do. Business Space can display the contents of one business space at a time. This space is the open space.

Before you can use the Business Space client to work with your human tasks, you must configure a user workspace. You use this workspace, and the available human task widgets, to claim and complete work items. You can also use Business Space to handle escalations and to work with the business calendars used by your human tasks.

It is important to use Firefox as the browser to view the Business Space content for this exercise. During the creation of this lab, the business space client was missing space styles if Internet Explorer was used.

1. Use the Business Space client to create a business space that is named: MyBusinessSpace

1. Start Firefox and enter `https://localhost:9443/BusinessSpace` in the location bar.
2. If the message is displayed that the connection is not secure then click Advanced, add exception and then confirm the exception.
3. At the **Login** prompt, enter `admin` in the **User ID** field and `web1sphere` in the **Password** field.

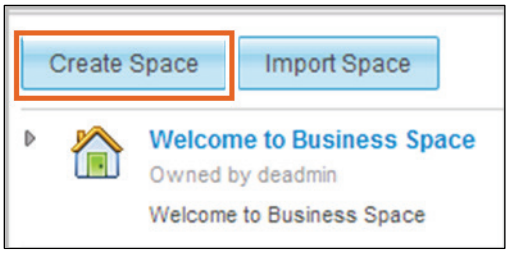


4. Click **Login**. Wait a few moments for the page to load.

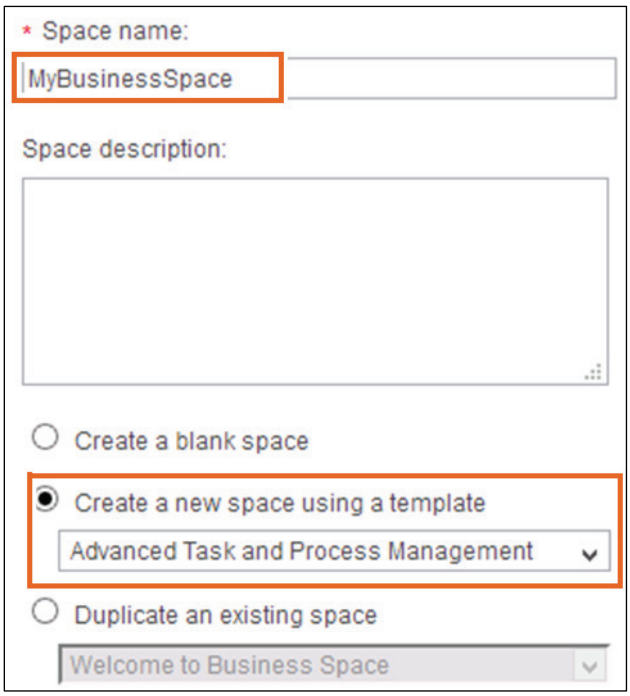
   Troubleshooting: During writing of this lab, sometimes, the Welcome page image did not display or render correctly. If that occurs, then it is OK to ignore the missing Welcome page, and you can continue with the exercise.

5. Click the **Manage Spaces** link in the action bar.

6. Create a workspace by clicking **Create Space**.



7. In the **Create Space** dialog box, take the following actions:
   - In the **Space Name** field, type: `MyBusinessSpace`
   - Click the **Create a new space using a template** option.
   - Select **Advanced Task and Process Management**.



With the preconfigured templates that come with Business Space, you can create a business space that contains pages and widgets. The template option lists the available templates within IBM Process Manager product. Two types of templates are available:

- Product templates have widgets from a single product. The Create Space window lists only the product templates for the products that are installed.
- Cross-product templates have widgets from more than one product.

In this exercise, you use the **Advanced task and Process Management** template to work on the assigned tasks.

Use the Advanced Management of Tasks and Processes template to create a space for business users. Team leads can also use this template. These team leads work on tasks, collaborate with other people, organize their own task lists, and check the status of processes and related tasks that they are involved with or are responsible for.

This template contains widgets from IBM Business Process Manager. It contains widgets and capabilities that are available only with the Business Process Choreographer engine. The widgets do not support the business process definition engine or federated process engines.