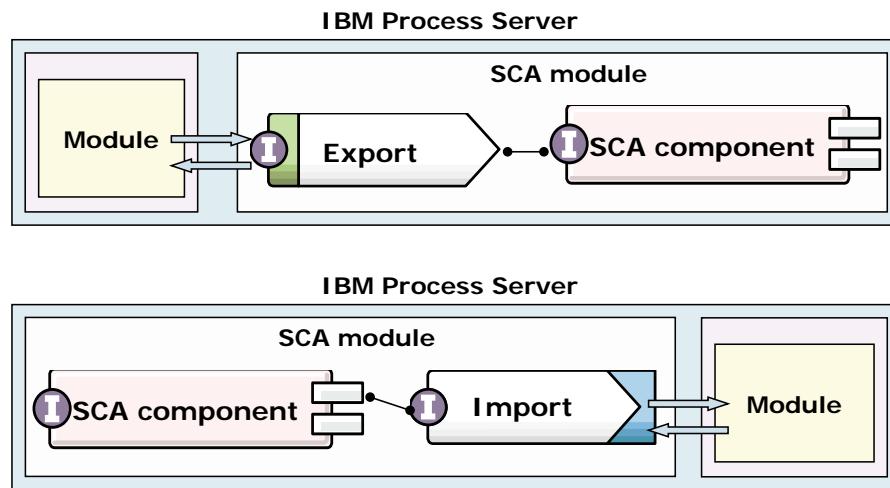


SCA bindings (2 of 2)

- If a service is exposed with an SCA bound export, another SCA module can invoke the service by using an SCA bound import component



SCA bindings

© Copyright IBM Corporation 2018

One of the capabilities of SCA is to describe an interface that can be called. The SCA runtime environment can decide how the interface call is made. If the calling code and the SCA module exist in the same runtime environment, it does not make sense to marshal or encode the data only to unmarshal it again when it reaches the destination.

The SCA runtime environment exposes an API that allows a caller to invoke an SCA-described service and, if the service is also exposed through SCA, to map the caller to the called code. An export component can be given an SCA binding, which allows a calling application in the same Java virtual machine to call that module as efficiently as possible. Examples of calling applications include EJB beans, JavaServer Pages (JSP pages), servlets, message-driven beans, and other Java EE artifacts.

JMS bindings (1 of 2)

- JMS is the Java Message Service specification, which describes how a Java application can send and receive messages
- Business value of JMS bindings:
 - Uses applications that are asynchronous or need assured delivery
 - Supports non-IBM JMS implementations
 - Easy integration with WebSphere MQ and messaging
 - Supports configurable message correlation (inbound and outbound)
 - Supports event sequencing for exports
 - Supports configurable reply connection for imports
 - Supports both publish and subscribe
 - Supports queue-based messaging
- Three types of JMS bindings:
 - JMS/MQ
 - JMS (WebSphere Application Server)
 - Generic JMS (for independent vendor JMS providers)

SCA bindings

© Copyright IBM Corporation 2018

JMS bindings

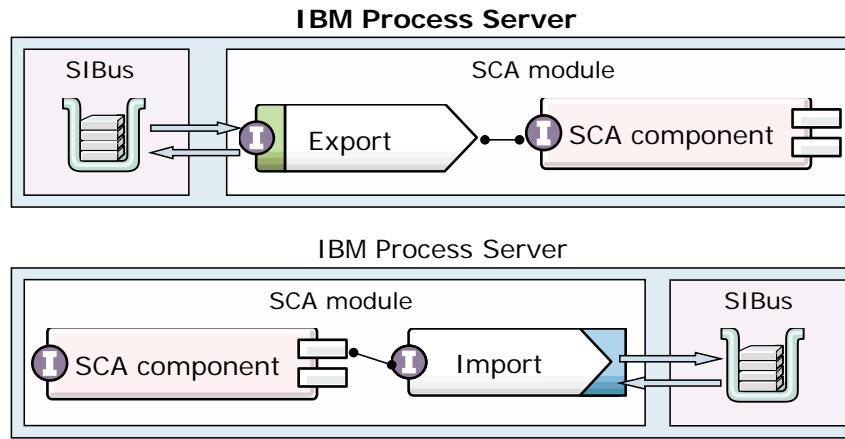
Java Message Service is a specification that describes how a Java application can send and receive messages. It supplies an API and semantics that are vendor independent. IBM implements JMS in the service integration bus (WebSphere Platform Messaging), IBM Process Server, and WebSphere MQ. If a Java application exists that is using JMS, IBM Process Server can send messages to it and receive messages from it. JMS binding supports the JMS message class plus all five JMS message subclasses: TextMessage, BytesMessage, ObjectMessage, StreamMessage, and MapMessage.

The JMS binding closely aligns with JMS/MQ binding capabilities. JMS bindings support:

- Configurable correlation schemes for both imports and exports
- Event sequencing for exports
- Configurable reply connection for imports

JMS bindings (2 of 2)

- Providing an export component with JMS binding allows module to watch an associated queue or topic and, when a message arrives, to receive the message and process it
- Providing an import component with JMS binding results in the production of a message when the service interface is called



© Copyright IBM Corporation 2018

JMS supports both point-to-point queuing and publish/subscribe messaging through topics. An SCA export with a JMS binding monitors an associated queue or topic; and, when a message arrives, the module receives the message and processes it. An SCA import can also have a JMS binding, which results in a message that is produced when the service interface is called.

Generic JMS bindings

- Generic JMS binding supports independent vendor JMS providers
 - Other web application servers
- Behavior is comparable to JMS and JMS/MQ bindings
 - Supports point-to-point and publish/subscribe styles
 - Supports correlation schemes and event sequencing
 - Security by using authentication aliases
 - Obeys SCA programming model
- Support is generic
 - No provider-specific connectivity options
 - Provider must be preconfigured
 - Limited capability to generate resources at deployment



SCA bindings

© Copyright IBM Corporation 2018

Generic JMS bindings

Generic JMS is a standard API for sending and receiving messages in a vendor-neutral manner. Generic JMS allows components to create, send, receive, and read messages regardless of the messaging system that is used, including non-IBM messaging providers. Generic JMS resources for WebSphere Application Server are automatically configured. Manual setup is required for independent vendor JMS resources.

WebSphere MQ can be used in IBM Process Server in multiple ways. They include WebSphere MQ bindings, MQ/JMS bindings, WebSphere MQ Link, and generic JMS bindings. It is preferable to use the WebSphere MQ bindings or the MQ/JMS bindings because they achieve the best synergy and performance.

WebSphere MQ bindings (1 of 2)

- WebSphere MQ:
 - Is a message-oriented middleware product
 - Provides reliable, resilient application integration by passing messages between applications and web services
 - Uses queuing and transactional facilities that help preserve the integrity of messages across the network
- Business value of bindings:
 - Provides easy integration with WebSphere MQ, WebSphere Application Server, WebSphere MQ Workflow, and IBM Integration Bus
 - Exports with WebSphere MQ binding allow external applications to drive execution of an IBM Process Server application

SCA bindings

© Copyright IBM Corporation 2018

WebSphere MQ bindings

Consider a WebSphere MQ binding when these factors are applicable:

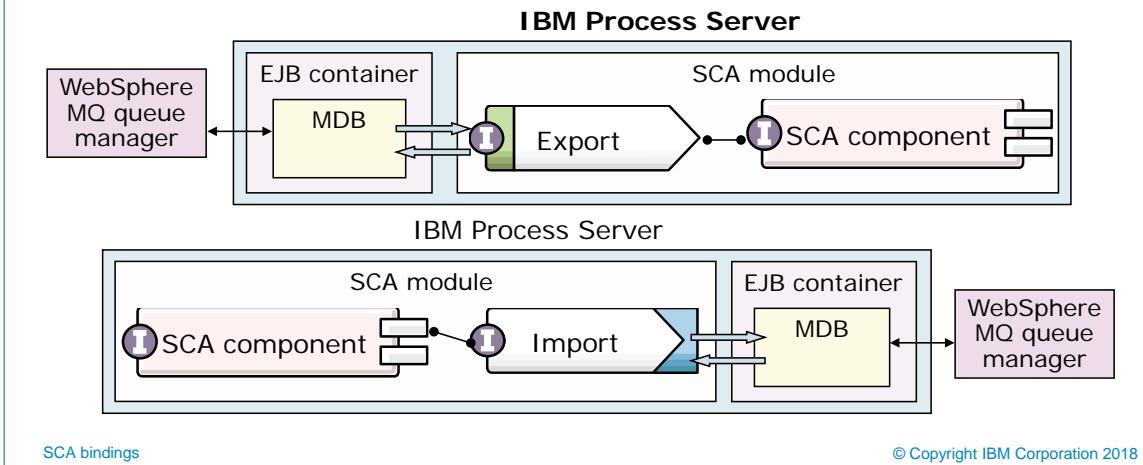
- You must access a WebSphere MQ messaging system, and you must use the WebSphere MQ basic functions.
- The services are loosely coupled.
- Reliability is more important than performance; that is, asynchronous data transmission is preferred over synchronous.

Consider an MQ/JMS binding when these factors are applicable:

- You must access a WebSphere MQ messaging system but can do so within a JMS context; that is, the JMS subset of functions is good enough for your application.
- The services are loosely coupled.
- Reliability is more important than performance; that is, asynchronous data transmission is preferred over synchronous.

WebSphere MQ bindings (2 of 2)

- When a message arrives on a queue, IBM Process Server monitors that queue (by using a message-driven bean)
- An import component with a WebSphere MQ binding provides an IBM Process Server module with the capability to drive the execution of an external application by sending a message to a queue



WebSphere MQ export binding allows an external WebSphere MQ application to drive the execution of an IBM Process Server application. When a message arrives on a WebSphere MQ queue, IBM Process Server can be configured to monitor that queue. WebSphere MQ import binding allows an IBM Process Server application to drive the execution of a WebSphere MQ application by sending a message to a queue.

On the Propagation tab in the Bindings section of the Properties view, you can enable the Propagate operation name option for the binding. Use it to add an RFH2 header to the outbound message that identifies the operation that is associated with the message.

The WebSphere MQ JCA resource adapter supports the handling of basic WebSphere MQ headers, which is not possible with the JMS API and the old WebSphere MQ JMS provider in prior versions.

HTTP bindings

- HTTP bindings use HTTP or HTTPS transport protocols to provide connectivity to an SCA-based application
- Business value:
 - Clients can reuse and build upon existing HTTP infrastructures
 - Brings benefits of SOA to HTTP deployments and allows SOA applications to take advantage of widespread HTTP applications (including Web 2.0 applications)
- HTTP bindings for imports and exports support:
 - HTTP 1.0 – 1.1 and SSL over HTTP
 - Synchronous request/response invocation only
 - Binary, XML, and SOAP payloads (plus custom data bindings)
 - Specifiable content and transfer encodings
 - Custom HTTP methods in imports
 - Endpoint-based routing in exports
 - Ability to modify the HTTP binding attributes at run time

SCA bindings

© Copyright IBM Corporation 2018

HTTP bindings

Enabling the HTTP binding in the business integration world helps you to easily compose HTTP-based services and to enhance the accessibility, performance, and scalability of your SCA applications. You can use the binding to allow an SCA application to use a wide variety of HTTP-based applications that exist on the World Wide Web. You can also create endpoints for HTTP clients, including the ones that can run in a simple web browser. Finally, you can more easily integrate your business integration applications with new emerging Web 2.0 technologies.

What is the HTTP binding?

Hypertext Transfer Protocol (HTTP) is a widely used protocol for transferring information about the web. Though originally designed to publish and retrieve HTML pages, it has now become a standard request/response protocol between clients and a server. It is defined according to the HTTP protocol that the World Wide Web Consortium (W3C) publishes. Today, many standard HTTP methods such as GET, POST, PUT, and DELETE are a part of this widely used protocol.

When working with an external application by using the HTTP protocol, an HTTP binding is necessary. The HTTP binding handles the transformation of data that is passed in as a message in a basic format to a business object in an SCA-based application. The HTTP binding also can transform data that is passed out as a business object to the basic format that an external application expects for an incoming message.

Why has the HTTP binding grown in importance?

When the SOAP/HTTP (referred to as SOAP over HTTP) web services binding gained momentum, businesses began seeing the potential of the HTTP protocol for business needs and application integration. SOAP/HTTP web services became the most common means for business-to-business transactions over the Internet. However, adding more robustness to SOAP/HTTP to suit business needs led to interoperability problems among vendors and more complexity of the initial specification. These robust services included the Web Service Description Language (WSDL), supporting the XML schema specification, and adding quality of service (QoS). It also led to the need for more skills to use SOAP/HTTP.

Users realized that many situations did not require the extra functions added to SOAP/HTTP. They merely wanted to take advantage of the ubiquitous web infrastructure to send or receive relatively simple information by using the HTTP protocol. They started by using HTTP-based services to send and receive data in a set of different loosely defined formats.

The HTTP binding is suited for this type of user as it combines the ease of use and simplicity of the original HTTP protocol with integration to large, scalable, and secure SOA applications. Conversely, the HTTP binding allows SOA applications to take advantage of the many existing HTTP-based applications, bringing them into the SOA framework. The binding also provides access from an SOA application to services that conform to the Web 2.0 specification. In summary, the HTTP binding allows applications that are developed for IBM Process Server to communicate with and mediate between the many web services by using HTTP and other protocols. This communication capability will make this binding only more important over time.

Comparing the HTTP binding to the web services binding

Another binding, the web services binding, also can be used with applications by using the HTTP protocol. The difference between the web services binding and the HTTP binding is as follows:

- The web services binding supports the SOAP (and JMS) protocols only.
- The web services binding assumes that it is working with web services-based applications and so exposes the same model. In contrast, the HTTP binding assumes that it is working with basic HTTP applications and exposes a model more familiar to this audience.
- Therefore, the web services binding provides first class support within the SCA architecture for web services applications that communicate with the HTTP protocol and other protocols. In contrast, the HTTP binding allows IBM Process Server to mediate between, and communicate with, any HTTP application, thus bringing any HTTP application into the service-oriented architecture framework.

HTTP binding at run time

In IBM Integration Designer, the HTTP binding can be used on imports and exports. An import with an HTTP binding at run time sends a request with or without data in the body of the message from the SCA application to the external web service. In other words, the request is made from the SCA application to the external web service. Optionally, the import with the HTTP binding can receive data back from the web application in a response to the request.

With an export, a client application to a web service makes the request. The web service is a web application that runs on the server. The export is implemented in that web application as a servlet so the client sends its request to a URL address. The servlet passes the request to the SCA application in the runtime environment. Optionally, the export can send data to the client application in response to the request.

HTTP bindings use HTTP-centric features. Messages are presented to SCA or mediation components in a manner that preserves HTTP protocol and message header information, which provides a more familiar view to HTTP application programmers, users, and administrators.

HTTP imports and exports can be configured to support a range of common HTTP features. They support HTTP 1.0 and 1.1 protocols and HTTPS, which uses SSL for secure HTTP conversation. The supported request/response mechanism is synchronous only. The binding supports static and dynamic HTTP headers (dynamic header setting access is done through mediation modules).

An existing DataBinding framework is extended for HTTP conventions and provides mapping between SCA messages and HTTP message headers and bodies. The IBM supplied data bindings support several different data payloads. The supported payloads are XML, binary, and SOAP. Users can create their own custom data bindings.

You can use the HTTP import to select the HTTP method, version, and other configuration. You can use the HTTP export to specify the endpoint URL for the client to invoke the services.

When the application with HTTP binding is deployed on the server, the administrator can modify many of the HTTP binding attributes by using the administrative console or the `wsadmin` commands.

The HTTP binding handles the transformation of data that is passed in as a message in a basic format to a business object in an SCA-based application. The HTTP binding also can transform data that is passed out as a business object to the basic format that the external application expects.

Web services

SCA bindings

© Copyright IBM Corporation 2018

Web services

Overview: Web services

- Web services connect businesses to each other and invoke services with appropriate security, reliability, and confidentiality
- If XML defines a platform-independent standard way to represent data, then web services define a platform-independent exchange for data
 - Application integration becomes easier
 - Web services use core technologies: XML, WSDL, and SOAP
- XML (Extensible Markup Language)
 - XML solves the problem of data independence
 - Use XML to describe data and to map that data into and out of any application
- WSDL (Web Services Description Language)
 - XML-based language to create a description of an underlying application
 - The description turns an application into a web service by acting as the interface between the underlying application and other web-enabled applications
- SOAP
 - SOAP is the core communications protocol for the web
 - Most web services use this protocol to communicate with each other
 - SOAP can be used over HTTP or JMS

SCA bindings

© Copyright IBM Corporation 2018

Overview: Web services

Web services are the common denominator across the industry, and virtually all environments support them. It is highly likely that users of IBM Process Server have web service-based components that they want to integrate. Web services are the common denominator for interoperation because they are standardized, supported by all middleware vendors and environments, and consistent across platforms and architectures.

Web services deliver key business value:

- Platform and language neutral
- Transport and data independent
- Designed to work well with the web (HTTP)
- Foundation for service-oriented architecture (SOA)
- Allow reuse of existing applications and infrastructure
- Ease of integration with IBM Integration Designer and IBM Process Server
- Easily choreographed in business processes

WSDL is an XML document format for describing web services as a set of endpoints that operate on messages that contain either document-oriented or procedure-oriented (RPC) messages. A WSDL document contains several parts:

- Types elements define custom data types for the service.
- Message elements declare message names and parameters.
- PortType elements list available operations.
- Binding elements describe supported protocols.
- Service elements map bindings to specific network addresses.

Discovering web services

- You typically discover service descriptions through a Universal Description, Discovery, and Integration (UDDI) registry or through WebSphere Service Registry and Repository
- By using WebSphere Service Registry and Repository, you can store, access, and manage information about services
 - You can use this information to select, invoke, and reuse services
- You can use WebSphere Service Registry and Repository to store information about services in your systems or in other systems that you already use, plan to use, or want to track
 - An application can check WebSphere Service Registry and Repository before it invokes a service to locate the most appropriate service that satisfies its functional and performance needs
 - This capability helps make your deployment more dynamic and more adaptable to changing business conditions
 - You can access WebSphere Service Registry and Repository through mediation flows
 - You can use the external service wizard with WebSphere Service Registry and Repository

Discovering web services

WebSphere Service Registry and Repository is the master metadata repository for service interaction endpoint descriptions. A broad definition of “service” applies here. This definition includes traditional web services that implement WSDL interfaces with SOAP/HTTP bindings and a broad range of SOA services. These services can be described by using WSDL, XSD, and policies, but might use a range of protocols and be implemented according to various programming models.

The integration point for service metadata, WebSphere Service Registry and Repository, establishes a central point for finding and managing service metadata that is acquired from a number of sources. These sources include service application deployments and other service metadata and endpoint registries and repositories. An example is Universal Description, Discovery, and Integration (UDDI), where service metadata that is scattered across an enterprise is brought together to provide a single, comprehensive description of a service. Then, visibility is controlled and versions are managed. The proposed changes are analyzed and communicated. The usage is monitored, and other parts of the SOA foundation can access the service metadata. They can be confident that they found the copy of record.

WebSphere Service Registry and Repository does not manage all service metadata, and it does not manage service metadata across the whole SOA lifecycle. It focuses on a minimalist set of metadata that describes capabilities, requirements, and the semantics of deployed service endpoints. It interacts and federates with other metadata stores that play a role in managing the overall lifecycle of a service.

During the development stages of the business service lifecycle, WebSphere Service Registry and Repository is used to locate the copies of record of candidate service interaction endpoints or mediating intermediaries, and policies that govern the interactions.

In the other SOA lifecycle stages, WebSphere Service Registry and Repository is complementary with repositories that specialize in managing SOA artifacts during the development stages of the lifecycle. For example, a development artifact management system such as Rational ClearCase takes care of service and composite application building blocks. These building blocks can include source code, service interface declarations, software architecture models, or business process models that are under construction. A reusable asset manager and asset repository manage bundles of artifacts that are described assets according to the Reusable Asset Specification (RAS) standard. They implement governance processes that control the promotion of artifacts to assets and the approval process that is associated with them.

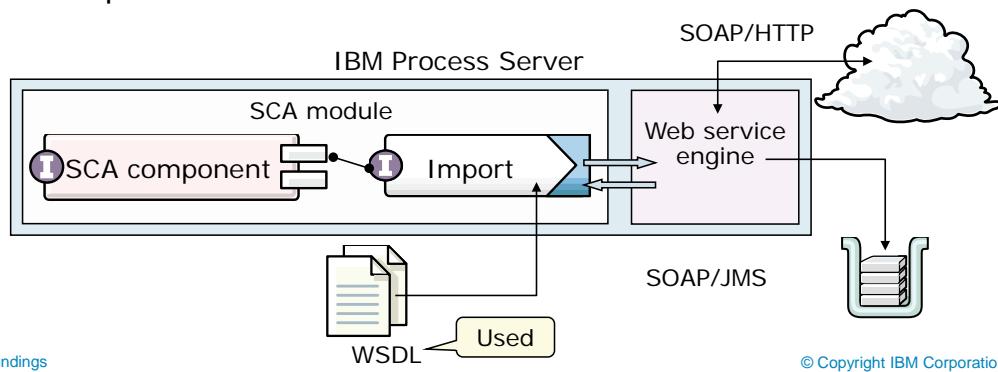
When the development team finishes its work and testing is complete, deployment teams further augment the service metadata, providing binding information for service endpoints that are used in composite applications. They manage deployment of the metadata from the development environment to the staging or production WebSphere Service Registry and Repository as part of the service deployment process.

Governance over the service metadata takes place, as metadata is promoted from test to staging to production environments that can have separate WebSphere Service Registry and Repository installations. In the production environment, WebSphere Service Registry and Repository is made available to a broader audience and shared. It is available to the runtime systems and those user roles that are responsible for the management of the IT systems.

Again, new service metadata, or more often a change in existing service metadata, can be discovered in other service endpoint registries and repositories. It is published in WebSphere Service Registry and Repository, and can be used as input for the application configuration and binding tasks that are the responsibility of deployment teams and solution administrators. Discovered service metadata is incomplete and not yet suitable for broader visibility and consumption. Deployment teams work with asset managers to ensure that the metadata is augmented with the necessary semantics, permissions, and scoping constraints.

Importing a web service WSDL interface

- When you discover and import a web service WSDL interface and drag it onto an assembly diagram in IBM Integration Designer, a web service bound import is created to call the service
- The component then refers to a web service endpoint
 - This endpoint is the location where the web service is listening for incoming requests
- An import with a web service binding results in a web service call to an external partner that uses SOAP over HTTP or SOAP over JMS



© Copyright IBM Corporation 2018

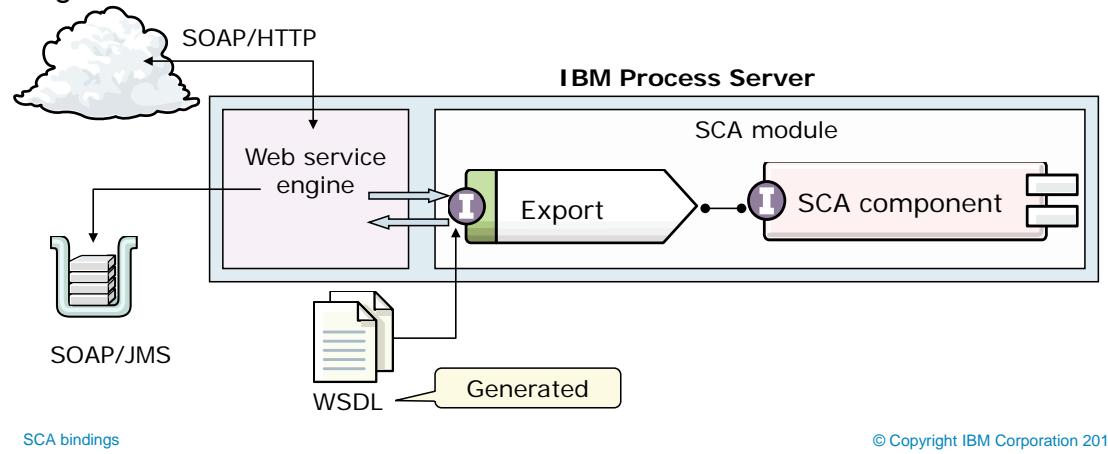
Importing a web service WSDL interface

An import component that is bound to a web service binding results in a web service call to an external partner when invoked. The request sent to the partner is either SOAP over HTTP or SOAP over JMS, depending on the binding request used. In this scenario, the external service that is called exists, so it is likely that a WSDL file describing that service is supplied.

When imported into IBM Integration Designer, the WSDL file can be directly inserted onto the SCA assembly diagram, causing the creation of “a ready-to-use web service bound import”. The WebSphere Application Server web service engine is used to build the request, send the request, and listen for the response.

Web service export binding (1 of 2)

- Exports with web service binding declare that the module exposes an interface that can be invoked remotely as a web service
- The export uses SOAP over HTTP or SOAP over JMS as the transport protocol
- When a web service export is declared, an associated WSDL file is generated



Web service export binding

Binding to web services is probably the most common of the binding types. An export component with a web service binding declares that the module exposes an interface, which can be invoked remotely as a web service. The export component can use either the SOAP over HTTP or the SOAP over JMS transport protocol. When a web service export is declared, an associated WSDL file is generated that can be given as-is to partners who want to invoke the module.

When a module with a web service export is deployed, the web service engine starts listening on behalf of the service. When a request arrives, this listener causes the instantiation of a module, and control is then passed to that module. The SCA runtime environment is a consumer of the web services engine and exposes SCA modules as web services through SOAP/HTTP.

Through the business object service, in-memory objects can be externalized as XML, or XSD contained in WSDL. This externalization is a convenient way of dealing with objects and XML directly. It also means that you can take Java objects and treat them as XSD.

The following IBM Process Server components are involved in web service communication:

- Service Component Architecture runtime environment
 - Client to the web service engine
 - Consumer of external web services
 - Architecture that is not centered on Java
 - Components that are exposed as SOAP and HTTP services
 - Client to “external” SOAP/HTTP services
- Business object service (in-memory)
 - Used by clients to access XSD/WSDL artifacts
 - Direct serialization between XML and Java (SDO)
- Web service engine
 - Part of the base WebSphere Application Server
 - JAX-RPC handlers (used for mediating SOAP messages) invoked for traditional bindings
 - Service integration bus

Web service export binding (2 of 2)

- Web service bindings support JAX-WS based web services
- JAX-WS bindings support:
 - WebSphere policy sets
 - SOAP 1.2 and SOAP 1.1
 - Attachments (unreferenced, referenced, and swaRef)
- JAX-WS 2.2 with MTOM support
- The older JAX-RPC web service binding is supported but is deprecated

Select a Transport Protocol

Select the transport protocol that CreditScoreServiceExport1 will use for the Web service binding.

<input checked="" type="radio"/> SOAP1.2/HTTP	JAX-WS bindings
<input type="radio"/> SOAP1.1/HTTP	Supports features such as JAX-WS handlers, WS-policy sets, and service gateways by using JAX-WS. More...
<input type="radio"/> SOAP1.1/HTTP using JAX-RPC	JAX-RPC bindings
<input type="radio"/> SOAP1.1/JMS	Supports JAX-RPC handlers and RPC encoded messages. More...

SCA bindings

© Copyright IBM Corporation 2018

IBM Integration Designer provides configuration options to enable JAX-WS bindings to send and receive web service messages, which include SOAP Message Transmission Optimization Mechanism (MTOM) attachments. This mechanism improves the transmission efficiency of large binary attachments in SOAP messages.

Before configuring a JAX-WS import or export binding to enable MTOM support, it is important to know that not all scenarios are supported:

- MTOM is not supported when the business object parsing mode is set to eager parsing (support is limited to a JAX-WS web service by using business object lazy parsing mode).
- MTOM is not supported when using a JAX-WS handler (support is limited to a JAX-WS web service, which does not use any JAX-WS handlers). The JAX-WS handlers that are specified on the web service should be removed.
- When using a service gateway mediation module, the Data Handler primitive cannot be used with MTOM messages. If direct access to the MTOM attachment data is required within the module, then a non-service gateway module must be used.

- When MTOM is enabled on a JAX-WS export binding, all responses are sent by using MTOM. If some clients do not support MTOM, use two JAX-WS exports, one with MTOM enabled and one with it disabled, and ensure that client applications use the correct endpoint address.
- MTOM is not supported when using the JAX-RPC binding for SOAP/HTTP or SOAP/JMS.

You can send and receive SOAP messages that include attachments that are represented in the service interface as `swaRef`-typed elements.

A `swaRef`-typed element is defined in the Web Services Interoperability Organization (WS-I) Attachments Profile Version 1.0 (<http://www.ws-i.org/Profiles/AttachmentsProfile-1.0.html>), which defines how message elements are related to MIME parts.

In the SOAP message, the SOAP body contains a `swaRef`-typed element that identifies the content ID of the attachment.

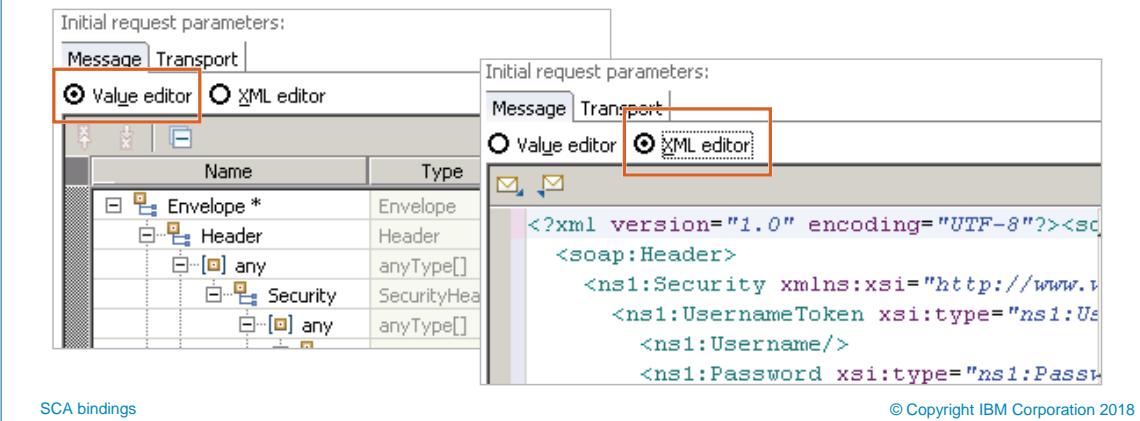
Policy sets reduce the complexity of configuring web services by providing reusable configurations.

A policy set is a collection of policy types, each of which provides a quality of service. These types are configured and can be associated with a web service provider or consumer. Policy sets work in pairs. You must have the same policy set on the service requester as on the service provider. Therefore, on an export, you should have the same policy set on the export binding as on the client. On an import, you should have the same policy set on the import binding as on the service provider it is calling.

A policy set can be associated with imports and exports. Only the SOAP1.2/HTTP and the SOAP1.1/HTTP transport protocols support policy sets. A policy set cannot be associated with the SOAP1.1/HTTP by using the JAX-RPC transport protocol.

Testing web services

- When you are testing web service exports, you can import data from XML or HTTP files
 - SOAP messages are stripped from the file
 - Input SOAP messages are editable
 - An HTTP file is captured from the TCP/IP Monitor or attached
- The integration test client has a full XML editor (value editor and XML editor) with syntax validation



Testing web services

In the integration test client, you can interactively send sample SOAP messages without attachments to a web service export and view the SOAP response.

Before testing a web service export with a SOAP message, you must have an existing SOAP message that you can use for testing. If you have HTTP messages that contain SOAP messages, they must be in *.http files. If you have pure SOAP messages with no attachments, the SOAP messages must be in *.xml files.

Unit summary

- List the various types of SCA import and export bindings
- Describe how SCA bindings facilitate integration with different types of applications
- Describe how web services are used in the Service Component Architecture framework

SCA bindings

© Copyright IBM Corporation 2018

Unit summary

Checkpoint questions

1. Define bindings in the context of SCA imports and exports.
2. What is the value of providing WebSphere MQ bindings for imports and exports?
3. What types of asynchronous invocation styles do SCA components support?
4. What is the difference between the JMS binding and the generic JMS binding?

SCA bindings

© Copyright IBM Corporation 2018

Checkpoint questions

Checkpoint answers

1. SCA binding information provides a consistent means for communicating across module boundaries in IBM Process Server. Bindings determine how your import and export components interact with clients outside a module. Bindings specify the protocol, message format, and invocation style.
2. It provides use for the huge installation base. It provides easy integration with WebSphere MQ, WebSphere Application Server, WebSphere MQ Workflow, and WebSphere Message Broker.
3. One-way, deferred response, and request with callback.
4. The JMS binding provides preconfigured resources that are optimized for WebSphere Application Server and IBM Process Server. Resources are automatically configured during deployment. Generic JMS does not contain preconfigured resources and does not create resources automatically during deployment.

Exercise 3: Working with web services

After completing this exercise, you should be able to:

- Import an external Web Services Description Language (WSDL) file into IBM Integration Designer
- Create an SCA component from a web service interface file
- Use the integration test client to test a web service
- Use a web service export to expose an existing IBM Process Server application

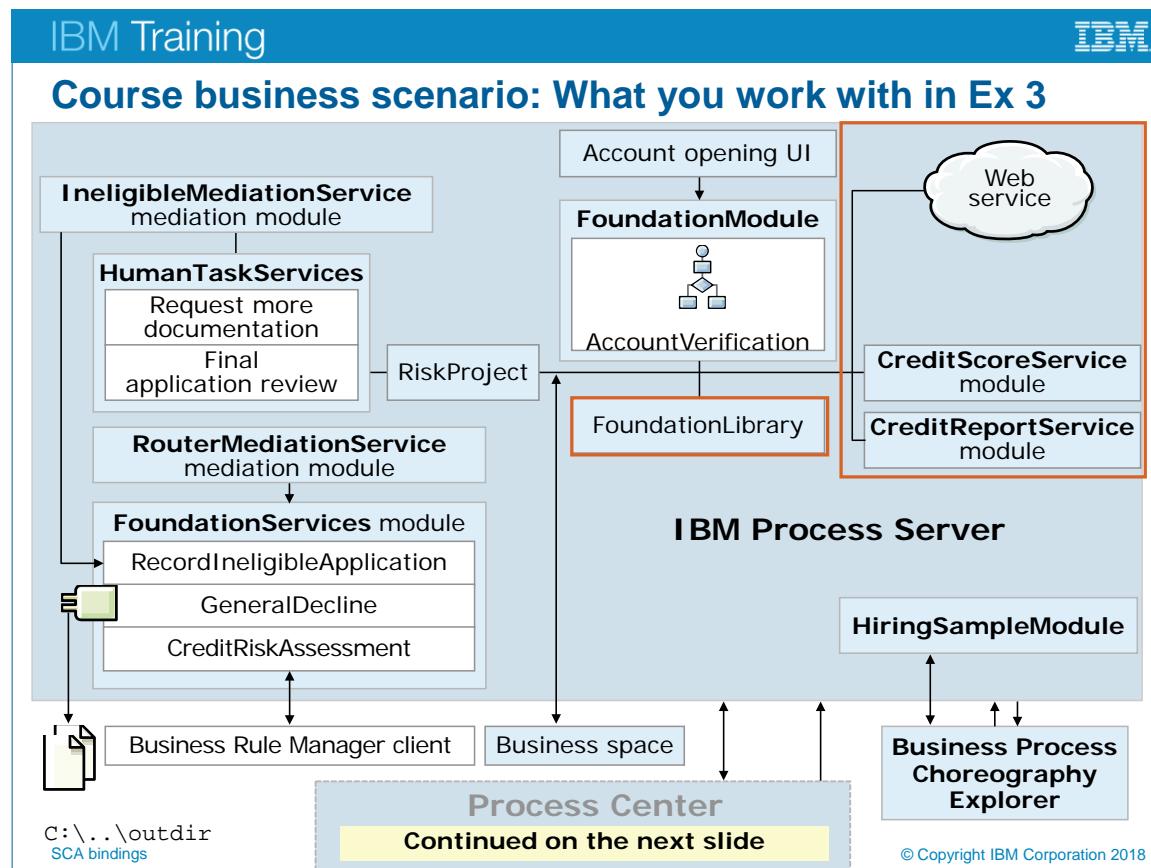
SCA bindings

© Copyright IBM Corporation 2018

Exercise objectives

After completing this exercise, you should be able to:

- Import an external Web Services Description Language (WSDL) file into IBM Integration Designer
- Create an SCA component from a web service interface file
- Use the integrated test client to test a web service
- Use a web service export to expose an existing IBM Process Server application



Course business scenario: What you work with in Ex 3

Components that are required for Exercise 3 (Section 1)

Prebuilt components that are imported in the lab:

1. CreditReportService.war

- Contains the **CreditReportServiceEAR** application
- You deploy the application to IBM Process Server

2. CreditReportService.wsdl

- Interface that describes the **CreditReportService** web service
- Defines an operation **CalculateCreditScore**
- Defines the **CreditCheckDetail** business object

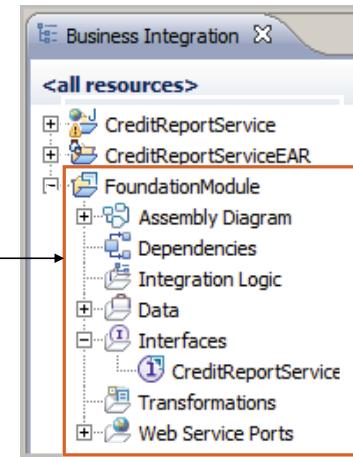
New components that you create in the lab:

1. FoundationModule

- Encapsulates and groups artifacts by type

2. CreditReportService

- SCA Import component that invokes the **CreditReportService** web service



© Copyright IBM Corporation 2018

Components that are required for Exercise 3 (Section 1)

Business Process Model and Notation, or BPMN, is a standardized graphical notation for creating diagrams of business processes.

BPMN is used so that everyone who is involved can interpret and understand the model. Throughout development, many different parties are involved in modeling. Every stakeholder, from the least technical to the most technical, understands the model to provide valuable feedback and continuously improve the process.

BPMN also allows a way to compact your process definition. Many of the symbols represent ideas, so symbols allow for a more concise and smaller model than drawing a diagram without BPMN.

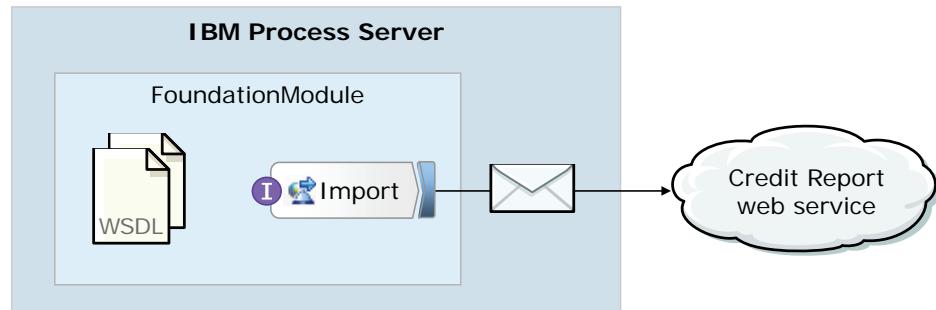
BPMN has many benefits, but most importantly, BPMN creates a standardized bridge for the gap between the business process design and process implementation. This single notation is agreed upon among multiple IBM BPM vendors for the benefit of the user community.

IBM implements and interprets these elements to have specific meanings and terminology in the IBM Process Designer product. For definitions of the BPMN specification, see the BPMN specification document version 2.0 from the Object Management Group. You can learn more about BPMN at: <http://www.bpmn.org>

The element palette of IBM Process Designer is shown in the slide image. These items correspond to BPMN elements and are used in modeling a process.

A BPMN standards group that is called Object Management Group is an international, open membership, not-for-profit technology standards consortium. You can search the Internet for this group.

Invoke an external web service in Exercise 3 (Section 1)



SCA bindings

© Copyright IBM Corporation 2018

Invoke an external web service in Exercise 3 (Section 1)

Components that are required for Exercise 4 (Section 2)

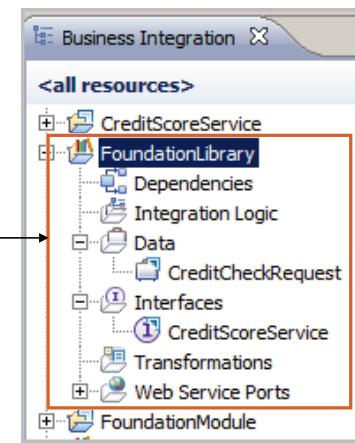
Prebuilt components that are imported in the lab:

1. **CreditScoreService**
 - Module containing the **CreditScoreRG** rule group that calculates the credit score
2. **CreditScoreService.wsdl**
 - Interface that describes the **CreditScoreService** web service
 - Defines the **CreditCheckRequest** business object

New components that you create in the lab:

1. **FoundationLibrary**
 - Container for shared business objects and interfaces:
CreditScoreService.wsdl interface and
CreditCheckRequest business object
2. **CreditScoreService**
 - Export SCA component that exposes the **CreditScoreService** web service
3. **CreditScoreService**
 - Import SCA component that calls the service through the SCA export

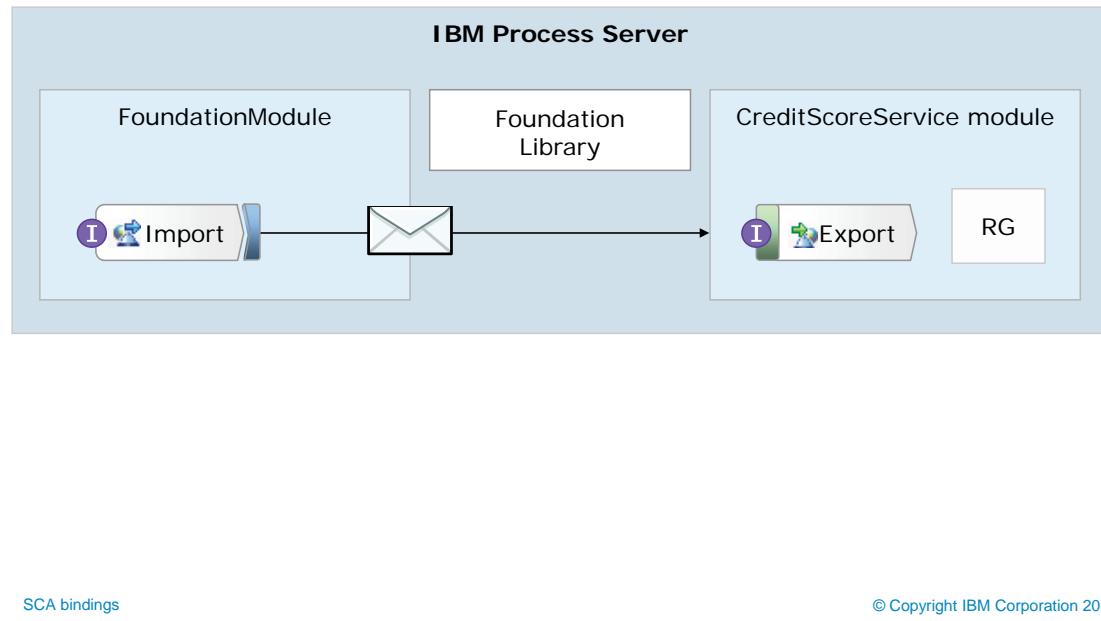
SCA bindings



© Copyright IBM Corporation 2018

Components that are required for Exercise 3 (Section 2)

Expose an application as a web service that is running in IBM Process Server in Exercise 3 (Section 2)



Expose an application as a web service that is running in IBM Process Server in Exercise 3 (Section 2)

Exercise 3: Working with web services

Purpose:

This exercise demonstrates two methods for working with web services in IBM Integration Designer. First, you import an external Web Service Description Language (WSDL) file given to you by a third party. You use this interface file to integrate a web service into your application by using IBM Integration Designer. After importing the WSDL file, you test the web service. Second, you use a web service export to expose an existing IBM Process Server application. You then use the SCA programming framework to call the export.

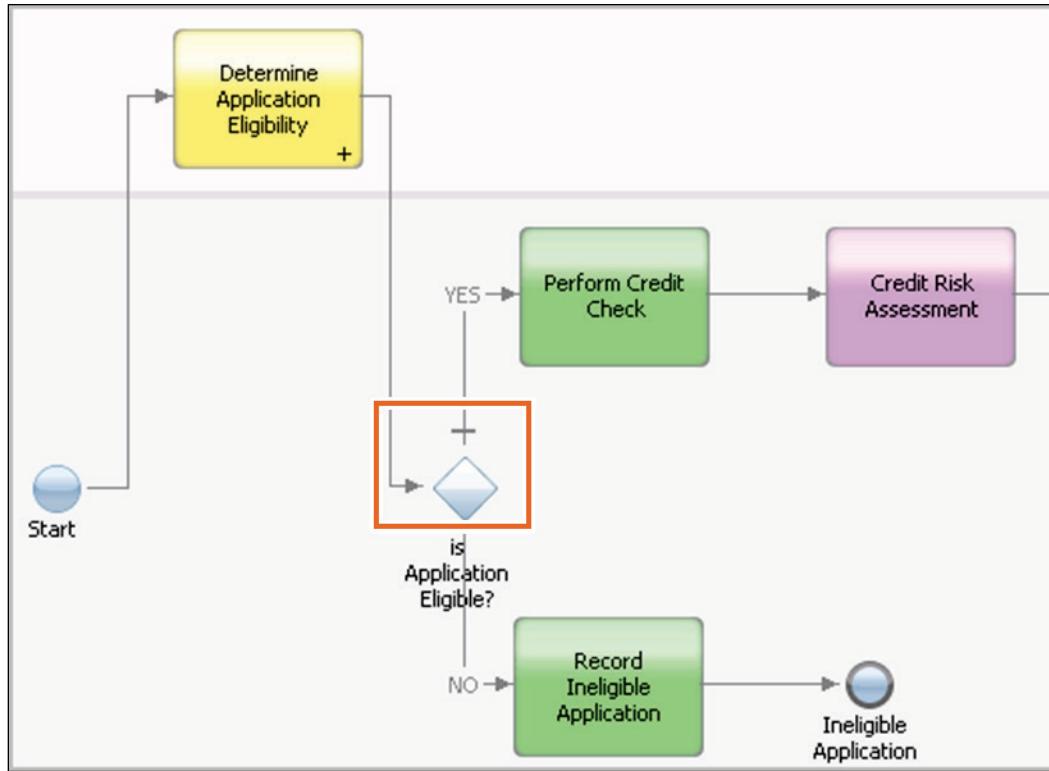
In your integration applications, you frequently invoke services that your Business Partners provide. The main advantage of web service technology is that it makes business-to-business communication simple and easy. IBM Integration Designer uses the Service Component Architecture (SCA) to facilitate the use of web service technology to connect to existing web services.

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

This exercise is based on the Account Verification process application, which you explored in IBM Process Designer in a previous lab.



As indicated in the account verification model and in the process narrative, when the customer submits an application, it must be tested for eligibility. If the application is eligible, the system calls an external service to do a credit check. In its implementation, this external service can realistically be a web service that another business provides to you.

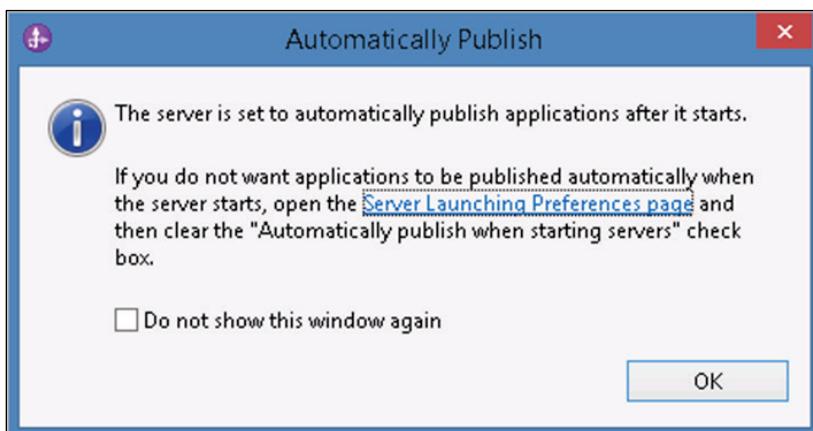
In this exercise, you use IBM Integration Designer to implement the Credit Check Service from this account verification model as a web service.

Part 1. Import an external Web Service Description Language file into IBM Integration Designer.

In this portion of the exercise, you import the WSDL interface of an external web service. In practice, a Business Partner is likely to offer this web service. After importing the WSDL interface, you use the IBM Integration Designer integrated test environment to examine the generated artifacts and test connectivity with the service.

1. Starting IBM Integration Designer and open the Ex3a workspace

1. On your desktop, open the folder that is labeled **Exercise Shortcuts**.
2. Double-click the shortcut that is labeled **Exercise 3a**.
Allow Integration Designer a few moments to build the workspace. You can view the workspace build status at the lower-right corner of the 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**. You get this message when the server is already running from the previous exercise.



4. Close the **Getting Started** tab.

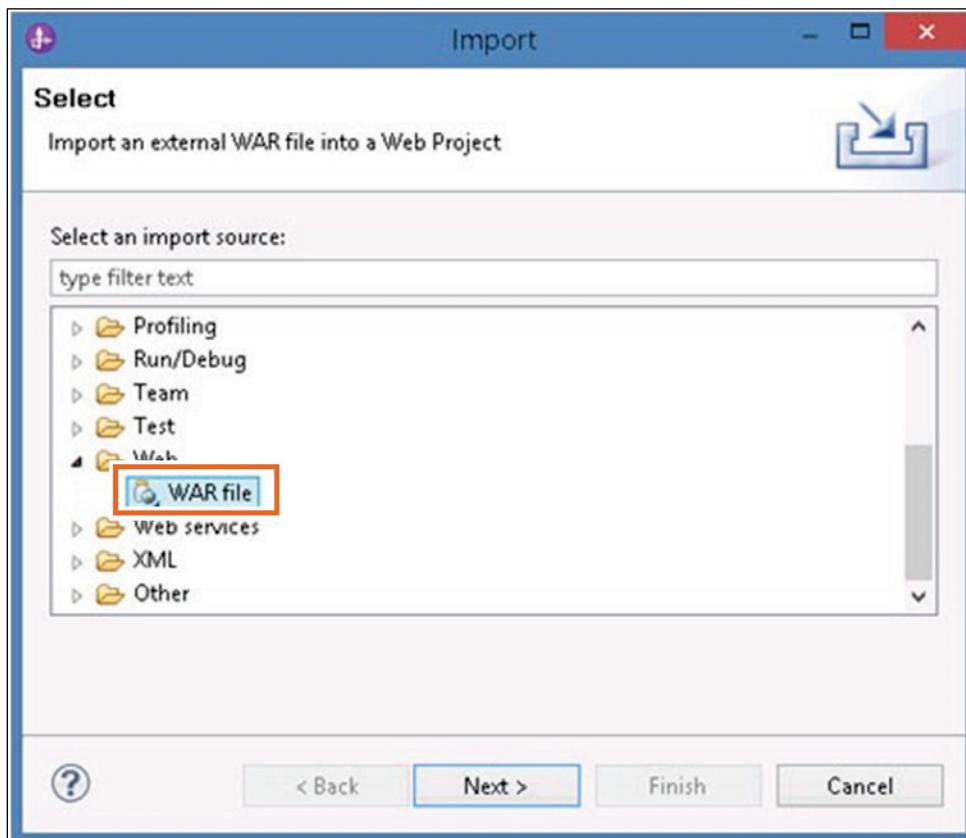
2. Starting the prebuilt web service.

Service-oriented architecture promotes flexibility and reusability of application functions. Based on the principles of service-oriented architecture, your Service Component Architecture application can interact with existing web services that other organizations, other companies, or your Business Partners provide.

Important: For simplicity, you are going to run a prebuilt web service, CreditReportService, on your local IBM Process Server. In a production environment, this web service would be running in your Business Partner's environment.

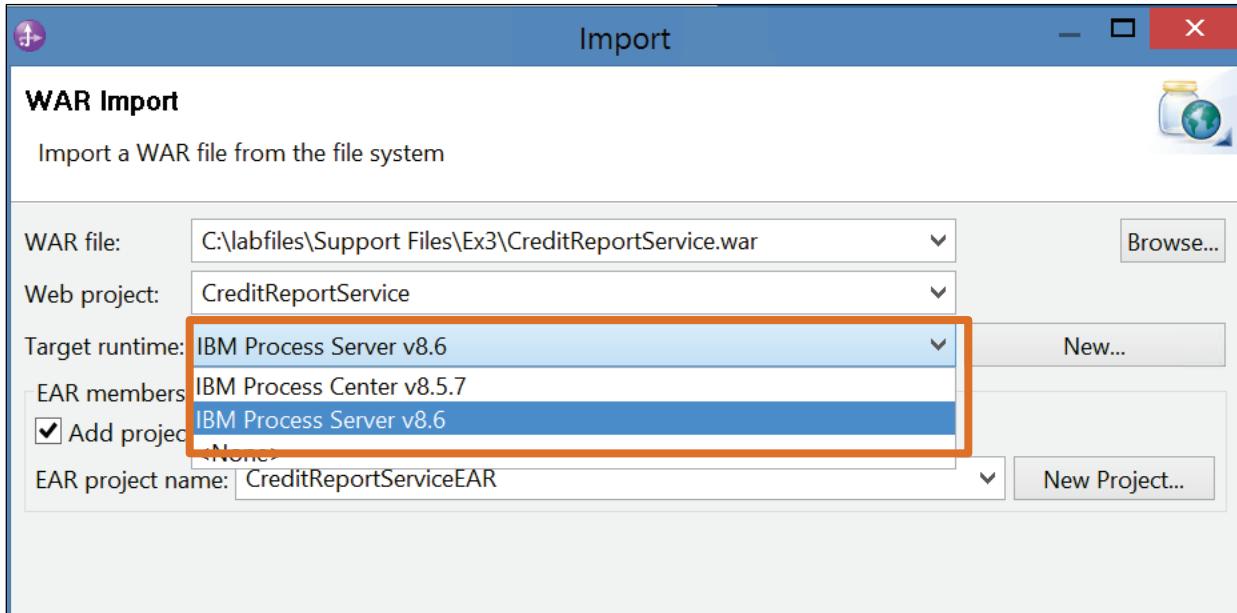
To start the web service, you import the CreditReportService.war file and start the web service.

1. Click **File > Import** from the menu options.
2. In the type selection window, expand **Web** and select **WAR file**.

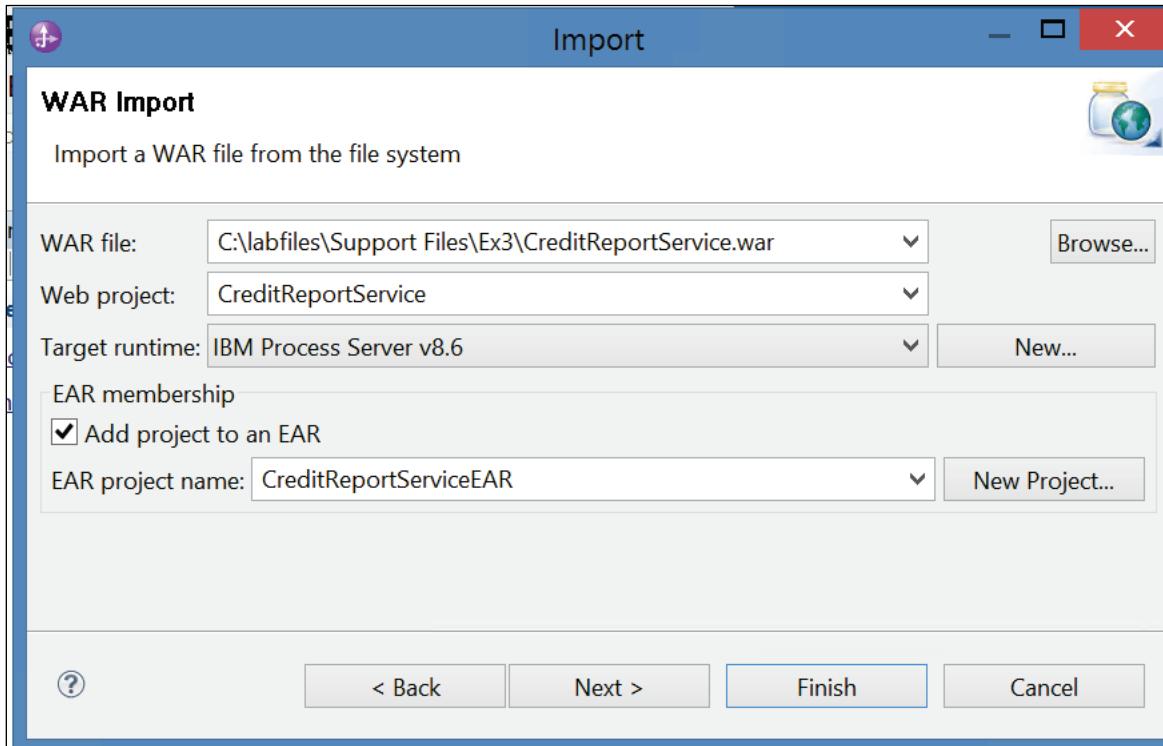


3. Click **Next**.

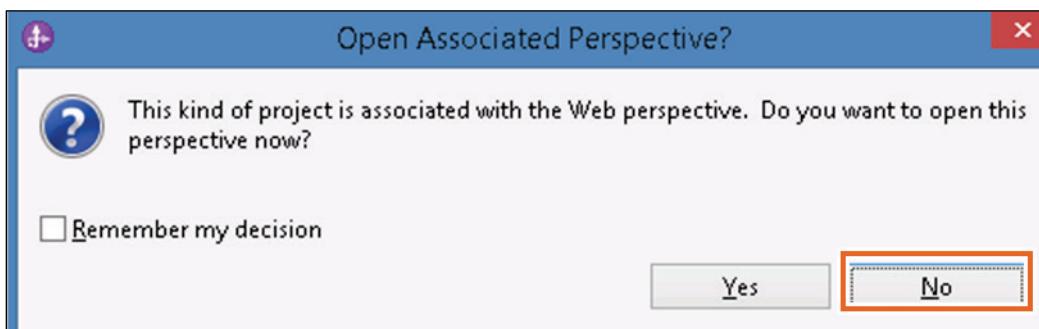
4. In the **Import** dialog box, click **Browse** beside the **WAR file** field and select **C:\labfiles\Support Files\Ex3\CreditReportService.war**.
5. Click **Open**. When you return to the WAR Import dialog box, in the **Target runtime** field, select **IBM Process Server v8.6**.



6. Accept the remaining default options and click **Finish**.



7. In the Open Associated Perspective dialog box, click **No** to avoid switching to the web perspective.

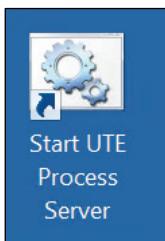


Information: Web services are not SCA artifacts, so it is necessary to switch to the appropriate perspective to work with these non-SCA components. IBM Integration Designer is built on top of Rational Application Developer technology.

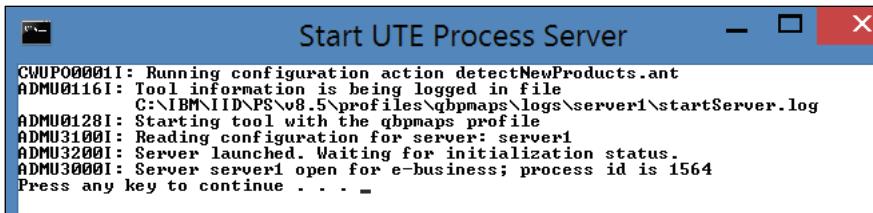
3. Start IBM Process Server.

If process server is already running from the previous exercise, then skip this step.

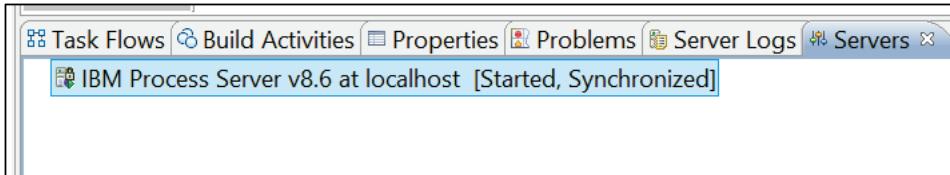
1. On your Windows desktop, select the **Start UTE Process Server** shortcut.



2. Double-click the shortcut or press Enter to start the server
3. A DOS command window is displayed; press any key to continue when prompted.

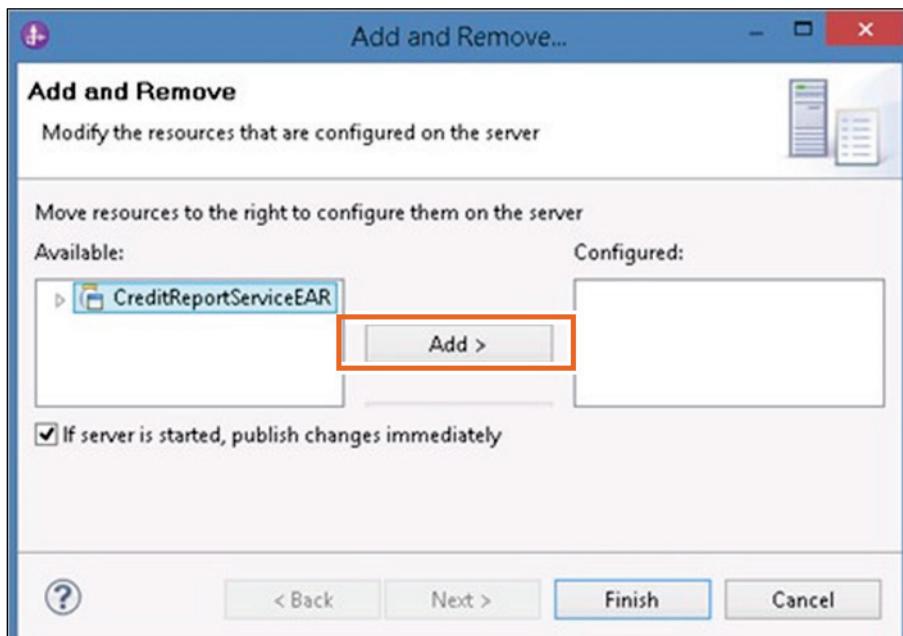


4. Return to Integration Designer and verify that the status of the server is **Started**.

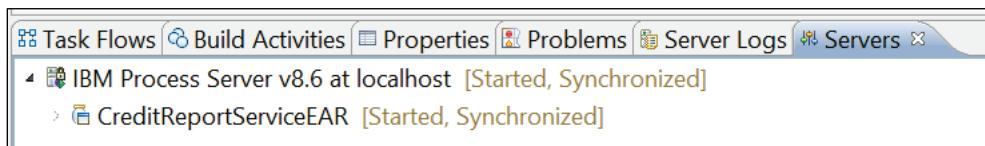


4. Add the EAR file to the server.

1. Right-click IBM Process Server v8.6 at localhost and click Add and Remove from the menu.
2. Select **CreditReportServiceEAR** in the Available list, and click Add.



3. When CreditReportServiceEAR is added to the **Configured** list, click **Finish** to deploy the CreditReportService web service.
4. Wait for the **CreditReportService** application to start. When the application is started, the following message is displayed in the **Server Logs** view:
Application started: CreditReportServiceEAR. The **Servers** view also shows the status of **CreditReportServiceEAR**.



5. If the **CreditReportServiceEAR** application has a status of **Stopped**, then right-click the module and click **Restart** in the menu. If prompted to republish the module, republish it.
6. 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**.

Part 2. Create a Service Component Architecture component from a web service interface file.

In this portion of the exercise, you import a WSDL file that describes the CreditReportService web service. Then, you test your connectivity with the service that is using the IBM Integration Designer test environment. When your connectivity is confirmed, you implement the SCA components that are used to connect to the service.

1. To import the WSDL interface, review the WSDL file that describes the CreditReportService web service.

1. In Windows Explorer, browse to C:\labfiles\Support Files\Ex3\.
2. Right-click **CreditReportService.wsdl** and click **Open with**.
3. Click the **More options** link in the window that opens and then double-click **WordPad**.
4. Examine the WSDL file.

CreditReportService.wsdl defines an operation **calculateCreditScore** that accepts an input message of type **calculateCreditScoreRequest**.

```
<wsdl:operation name="calculateCreditScore">
    <wsdlsoap:operation soapAction="calculateCreditScore" />

    <wsdl:input name="calculateCreditScoreRequest">
        <wsdlsoap:body use="literal" />
    </wsdl:input>
```

The **calculateCreditScoreRequest** message is associated with data type **calculateCreditScore**.

```
<wsdl:message name="calculateCreditScoreRequest">
    <wsdl:part element="intf:calculateCreditScore"
        name="parameters" />
</wsdl:message>
```

The **calculateCreditScore** data type is a complex data type that includes **CreditCheckDetail**.

```
<element name="calculateCreditScore">
    <complexType>
        <sequence>
            <element name="request" nillable="true"
                type="impl:CreditCheckDetail"/>
        </sequence>
    </complexType>
</element>
```

The definition of **CreditCheckDetail** contains the elements `acctNumber`, `companyName`, `creditScore`, and `dateRequested`.

```
<complexType name="CreditCheckDetail">
    <sequence>
        <element name="acctNumber" nillable="true"
            type="xsd:string"/>
        <element name="companyName" nillable="true"
            type="xsd:string"/>

        <element name="creditScore" type="xsd:int"/>

        <element name="dateRequested" nillable="true"
            type="xsd:string"/>
    </sequence>
</complexType>
```

Note: The web service address that you use to test the service is:

```
<wsdlsoap:address
location="https://localhost:9443/CreditReportService/
services/CreditReportService"/>
```

In a production environment, the address would contain

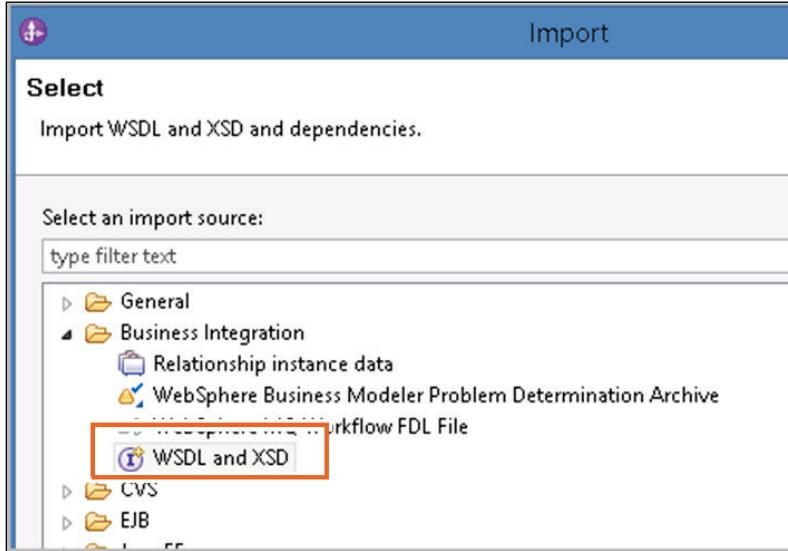
`https://www.<webaddress>.com/CreditReportService/` instead of
`localhost:port`.

5. Close `CreditReportService.wsdl`.
6. Close Windows Explorer.

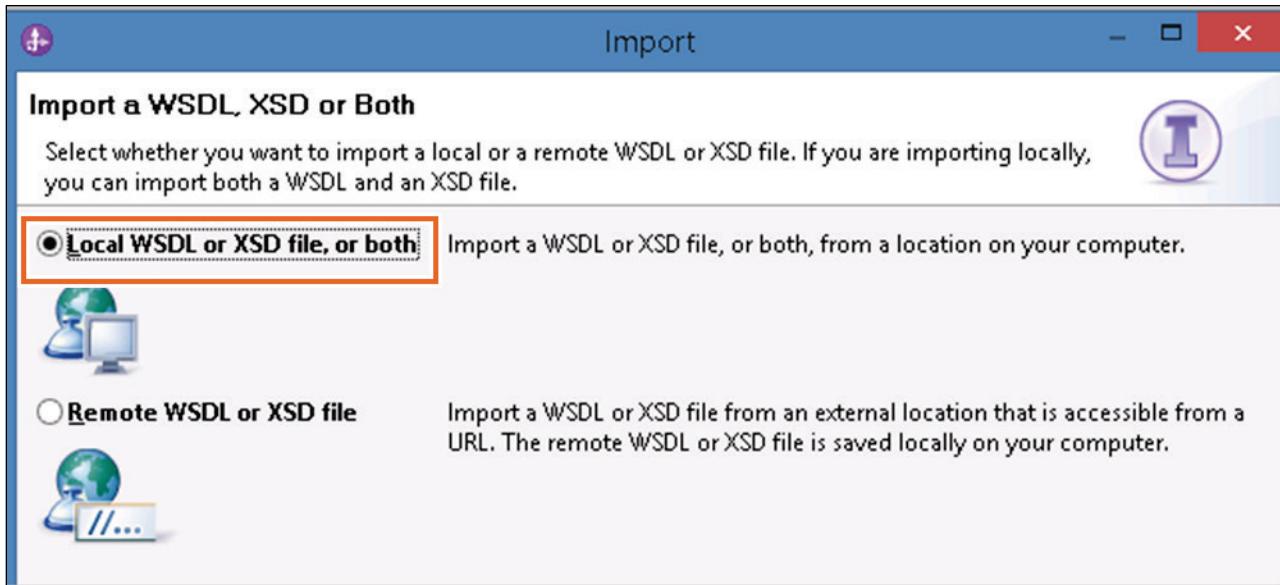
2. Create an IBM Integration Designer project that is called FoundationModule.

This module is the starting point for the new end-to-end application to develop.

1. Click **File > New > Module** from the menu options.
2. At the “Create a Module” dialog box, type `FoundationModule` in the **Module name** field.
3. Accept the remaining default options and click **Finish**.
4. Wait until no messages appear in the IBM Integration Designer status bar, such as `Building workspace`.
3. Import `CreditReportService.wsdl` into `FoundationModule`.
 1. In the Business Integration view, right-click **FoundationModule** and click **Import**.
 2. Expand **Business Integration** and select **WSDL and XSD**.

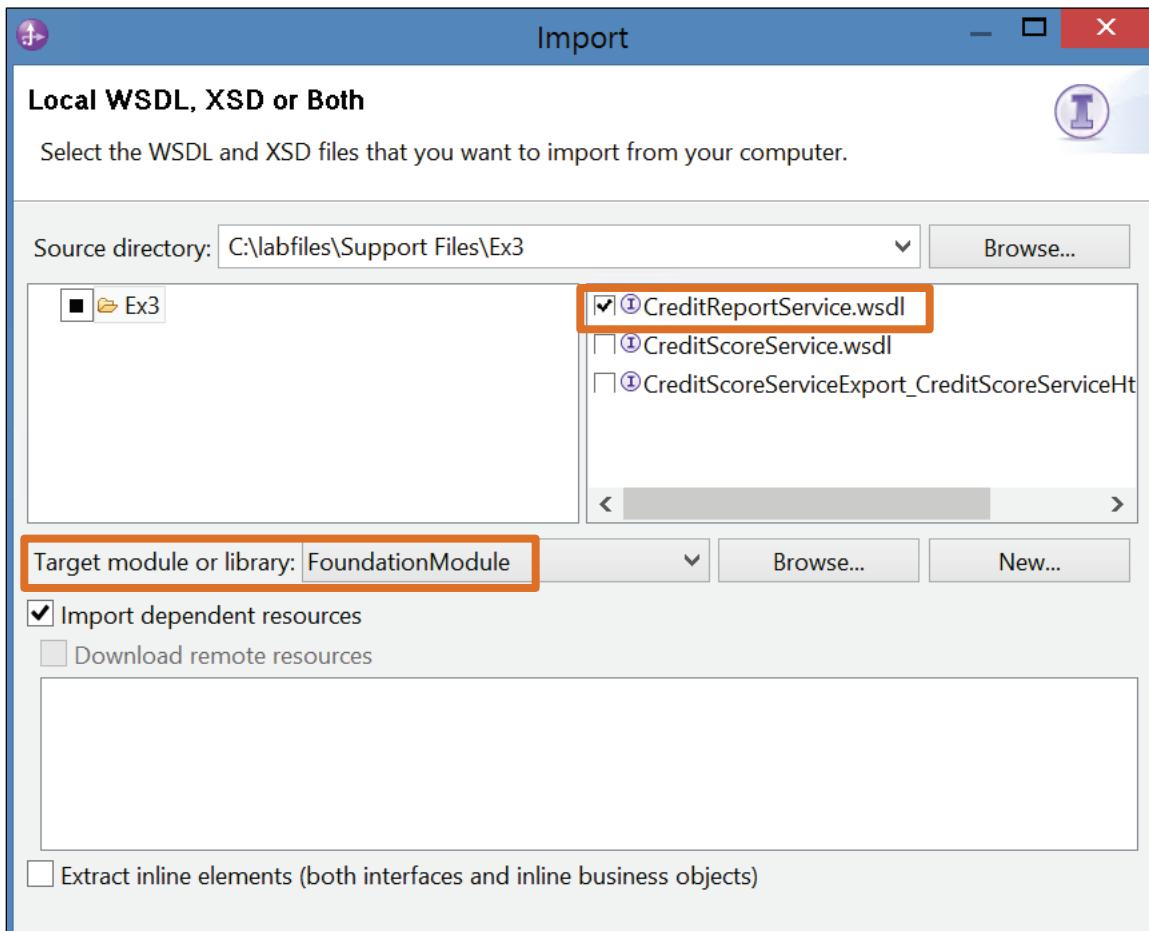


3. Click **Next**.
4. At the **Import a WSDL, XSD or Both** dialog box, select **Local WSDL or XSD file, or both**.



5. Click **Next**.
6. Click **Browse** next to **Source directory** and browse to:
C:\labfiles\Support Files
7. Select **Ex3** and click **OK**.
8. Select only the check box for **CreditReportService.wsdl**.

9. Verify that **Target module or library** is set to: FoundationModule



10. Accept the remaining default options and click **Finish**.

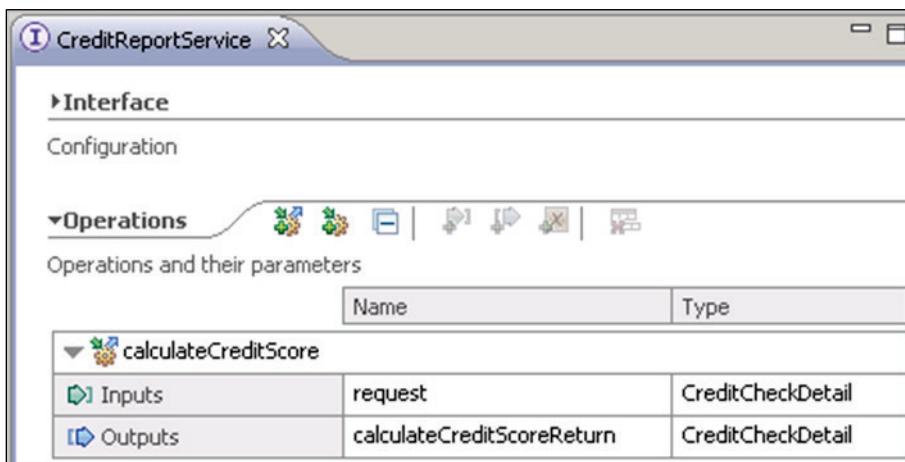
4. Examine the SCA artifacts in FoundationModule that were created as a result of the WSDL import.

These artifacts were generated based on the information you examined earlier in the WSDL file.

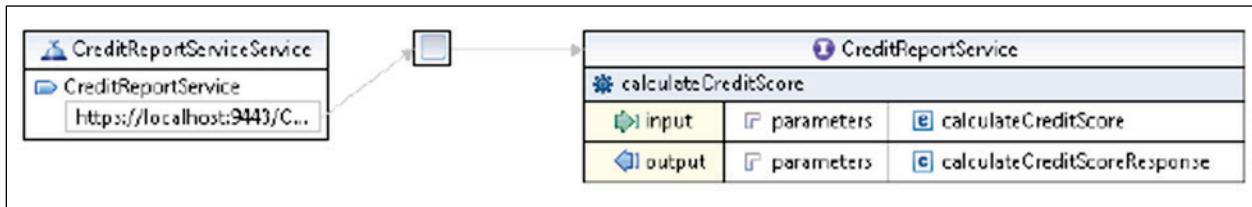
1. Expand **FoundationModule > Data** and double-click **CreditCheckDetail** to open the business object definition.



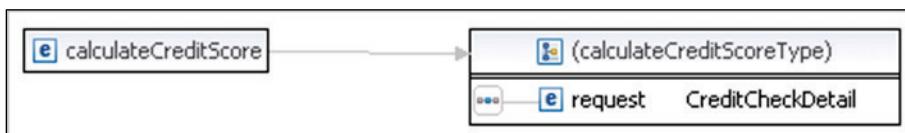
2. Expand **Interfaces** and double-click **CreditReportService** to view the interface definition.



3. Close the **CreditCheckDetail** tab and the **CreditReportService** tab.
 5. Examine the WSDL file in the WSDL editor.
The WSDL editor provides a graphical representation of the WSDL file that you examined previously in a text editor.
1. Expand FoundationModule > Interfaces.
 2. Right-click **CreditReportService** and click **Open With > WSDL Editor** from the menu.
- The WSDL editor provides you with a graphical view of the service address, the interface operation, and the input and output parameters. As you saw previously, the **calculateCreditScoreRequest** message is associated with data type **calculateCreditScore**.
3. Click the arrow to the right of **calculateCreditScore** input (it changes color when you hover over it).



4. A new tab opens, containing the inline schema of the WSDL file. The **calculateCreditScore** data type is a complex data type that includes **CreditCheckDetail**.



5. As time permits, explore other aspects of the interface in the WSDL editor. When you are done, close any open tabs.

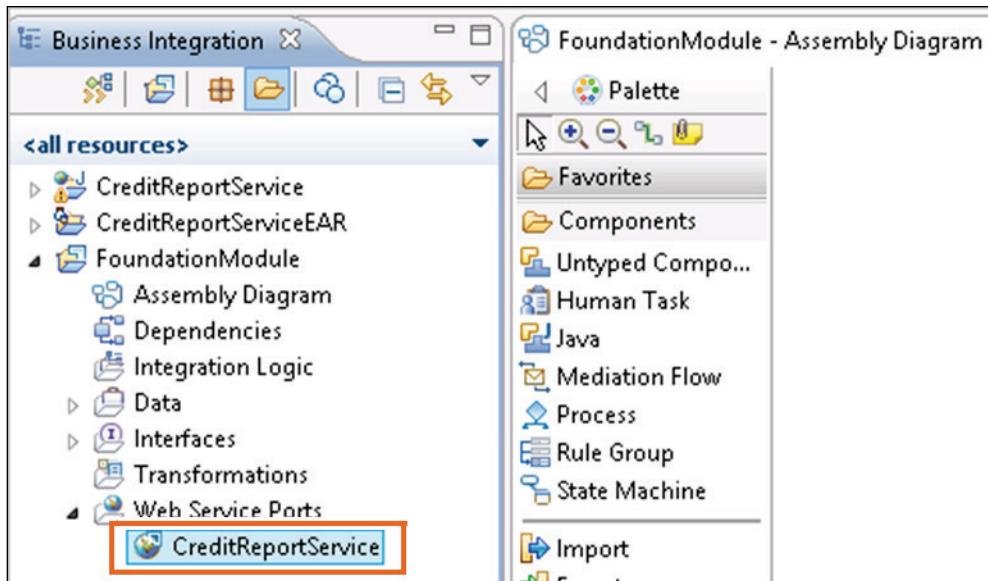
Part 4. Creating an import component to invoke the CreditReportService.

In the previous section, you imported the WSDL interface for your web service. In this portion of the exercise, you create the SCA import component that is used to invoke the CreditReportService web service in your application.

1. Create a CreditReportService import component.

In the section, you create an import component to invoke the **CreditReportService** web service.

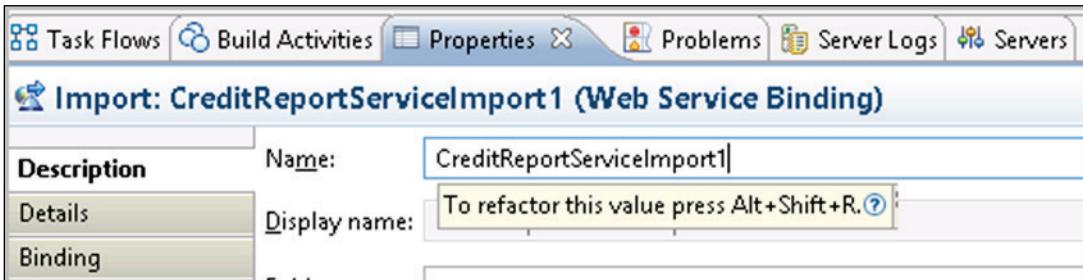
6. If the assembly diagram is not open, expand **FoundationModule** and double-click **Assembly Diagram**.
7. In the Business Integration view, expand **FoundationModule > Web Service Ports** and select **CreditReportService**.
8. Drag **CreditReportService** onto the assembly diagram.



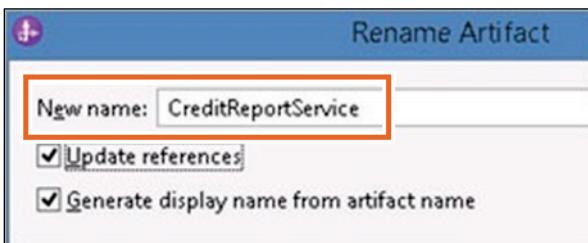
9. Select the **SOAP1.1/HTTP** option and click **Finish**.
10. Press **Ctrl+S** to save your changes.

2. Change the default name of the import.

1. Select the CreditReportServiceImport1 component.
2. Switch to the **Properties** view.
3. With the cursor in the **Name** field, press Alt+Shift+R to refactor the name.



4. In the Rename Artifact dialog box, type CreditReportService in the **New name** field.



5. Click **Refactor**.

You can also analyze the effect of the changes before you click Refactor.

Hint: Although you can type the new name without using refactor, it is always a good idea to get in the habit of using Refactor to rename your components. These components often are referenced in other modules or libraries. Using refactor changes the name in all locations where the component is used.

Part 5. Use the integrated test client to test a web service.

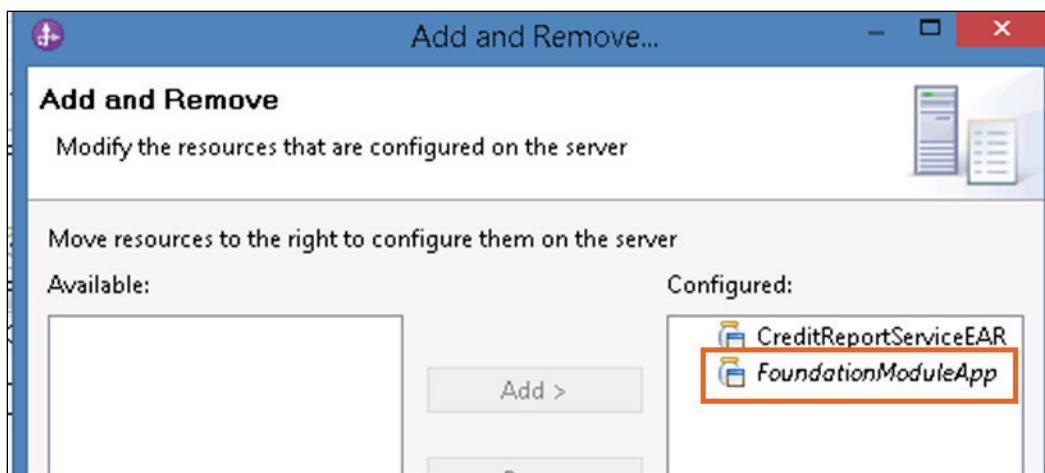
In this portion of the exercise, you test the web service import component. Using the IBM Integration Designer test client, you call the web service that is running in the local IBM Process Server test environment, which simulates a web service that is running in an external location. Many of the features in IBM Integration Designer support web services. A complete, full-featured web service development environment is available in Rational Application Developer.

1. Test the credit check service import.

You deploy **FoundationModule** to the server and test the **CreditReportService** import component.

1. If your server is not already started, start it by using the desktop shortcut.
Note: The server was already started when you deployed the credit check service EAR file earlier.
2. Right-click IBM Process Server v8.6 at localhost and click Add and Remove from the menu
3. Double-click **FoundationModuleApp** to add it to the **Configured projects** list, and click **Finish**.

CreditReportServiceEAR was deployed previously.



Wait until the module is published and started. Publishing is complete when no messages appear in the IBM Integration Designer status bar such as **Publishing FoundationModuleApp**. The application is started when you see the message **Application started: FoundationModuleApp** in the **Server Logs** view.

Type	Time	Thread ID	Contents
Log message	Apr 18, 2016 14:14:46.558 EDT	00000057	WSVR0200I: Starting application: CreditReportServiceEAR
Log message	Apr 18, 2016 14:14:47.073 EDT	00000057	WSVR0221I: Application started: CreditReportServiceEAR
Log message	Apr 18, 2016 14:14:51.511 EDT	00000057	WSVR0200I: Starting application: FoundationModuleApp
Log message	Apr 18, 2016 14:14:52.401 EDT	00000057	WSVR0221I: Application started: FoundationModuleApp

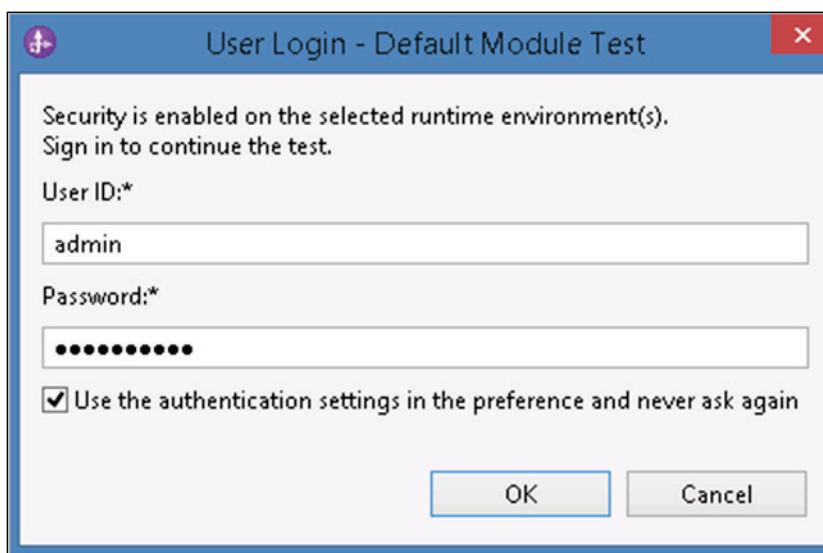
4. If the FoundationModuleApp application has a status of **Stopped**, then right-click the module and click **Restart** in the menu. If prompted to republish the module, republish it.
5. In the assembly diagram, right-click **CreditReportService** and click **Test Component** from the menu to open the integrated test client.

6. In the **Initial request parameters** table, enter the following test data:

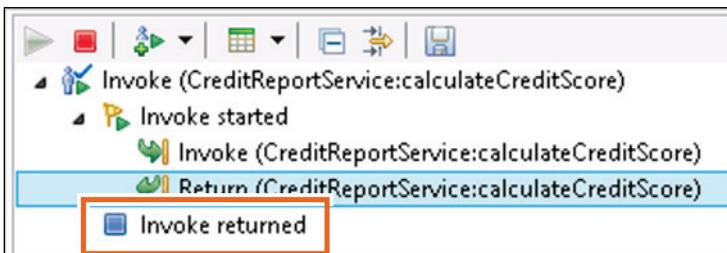
- acctNumber: 100
- companyName: IBM
- creditScore: 0
- dateRequested: 04/15/2016

Name	Type	
request	CreditCheckDetail	[ab]
acctNumber *	string	[ab] 100
companyName *	string	[ab] IBM
creditScore *	int	[ab] 0
dateRequested *	string	[ab] 04/15/2016

7. Click **Continue** on the Events toolbar.
8. When the Select a Deployment Location dialog box is displayed, select **IBM Process Server v8.6 at localhost**, select **Use this location as the default and do not ask again**, and click **Finish**.
9. When the User Login dialog box is displayed, select the **Use the authentication settings in the preference and never ask again** check box, and click **OK**.



- Wait for the test to complete. When the test is finished, a stop node labeled "Invoke returned" is listed in the Events window.

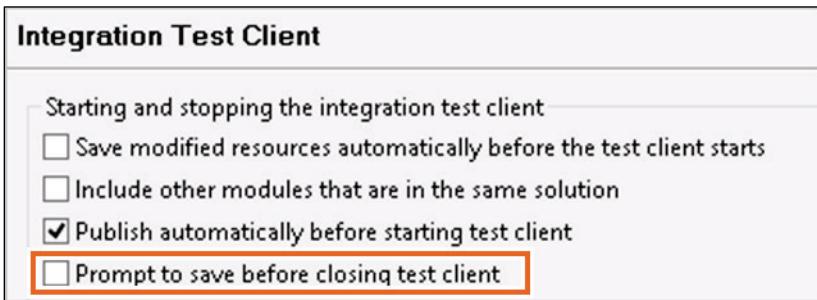


- Select the **Return** event unless it is already selected and examine the **Return parameters** section.
- The result of the test is a randomly generated credit score in the range 0–11. Because the score is generated randomly, your result might differ from the following screen capture.

Return parameters:		
<input type="button" value="Value Editor"/> <input type="button" value="XML Source"/>		
Name	Type	Value
calculateCredi...	CreditCheckDetail	[ab]
acctNumber	string	[ab] 100
companyNar	string	[ab] IBM
creditScore *	int	[ab] 10
dateRequeste	string	[ab] 04/15/2016

- Close the test client tab. When you are prompted to save the test trace, click **No**.

Note: You can disable the dialog box that prompts you to save the test trace. To disable the dialog box, click **Window > Preferences > Business Integration > Test > Integration Test Client**. Clear the **Prompt to save before closing test client** check box and click **OK**.



- Remove the projects and (optionally) stop the server.
- In the Servers view, right-click IBM Process Server v8.5.6 at localhost and click Add and Remove from the menu.

2. Click **Remove All** and click **Finish**. Wait until no messages appear in the IBM Integration Designer status bar.

Note: If your computer has ample memory or if you are working on the exercises in the remote environment (for instructor-led online or self-paced courses), you might want the server to continue running to save time. If you choose to leave it running, you can ignore any steps in subsequent labs that require you to stop the server.

3. Close IBM Integration Designer.

Part 6. Use a web service export to expose an existing IBM Process Server application.

In this portion of the exercise, you use a different approach to implementing web service connectivity. In this approach, you use an export with a Web Service Binding type to expose an existing IBM Process Server application. In future exercises, this application replaces the service that you used in the previous section.

Unlike the previous service, this web service returns a specific credit score value for each of the test cases that are built into your application. For more information about how the application was created, see the appendixes.

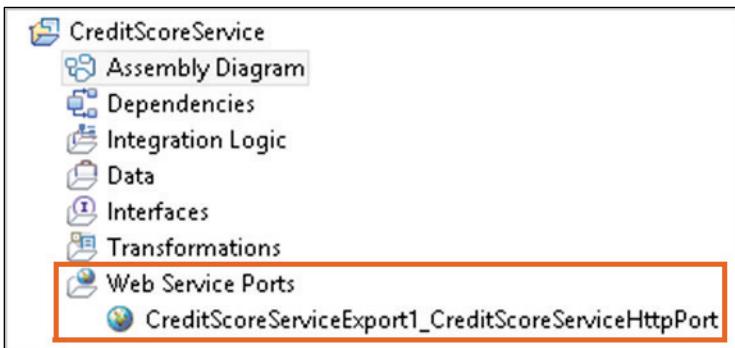
1. To expose an application by using a web service export.
 1. Open the **Exercise 3b** workspace.
 2. On your desktop, open the **Exercise Shortcuts** folder.
 1. Double-click the **Exercise 3b** shortcut.
 2. If you get a message that the server is already set to publish, then click **OK**. You get this message when the server is already running.
 3. Close the **Getting Started** tab.
2. Use an export component with a Web Service Binding to expose the credit score rule group.
 1. In the Business Integration view, expand **CreditScoreService** and double-click **Assembly Diagram**.
 2. Right-click the **CreditScoreRG** component and click **Generate Export > Web Service Binding** from the menu.

- In the “Select a Transport Protocol” dialog box, select **SOAP1.2/HTTP**. Because you are generating a new Web Service Binding, you can use the newer SOAP V1.2 protocol.

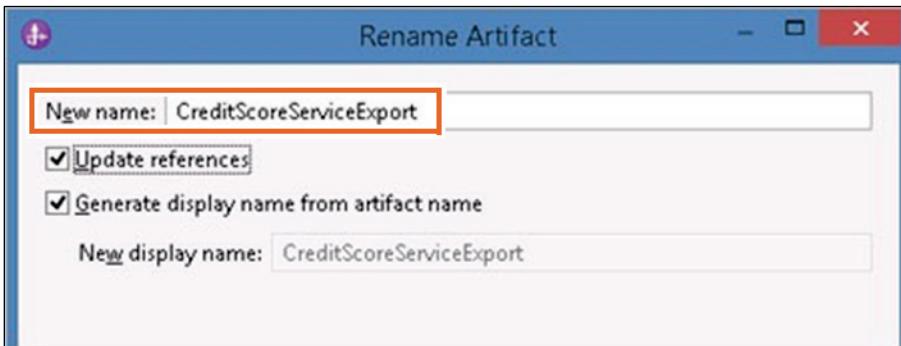


- Click **Finish**.

When the export is generated, a **Web Service Ports** section is added to the project.



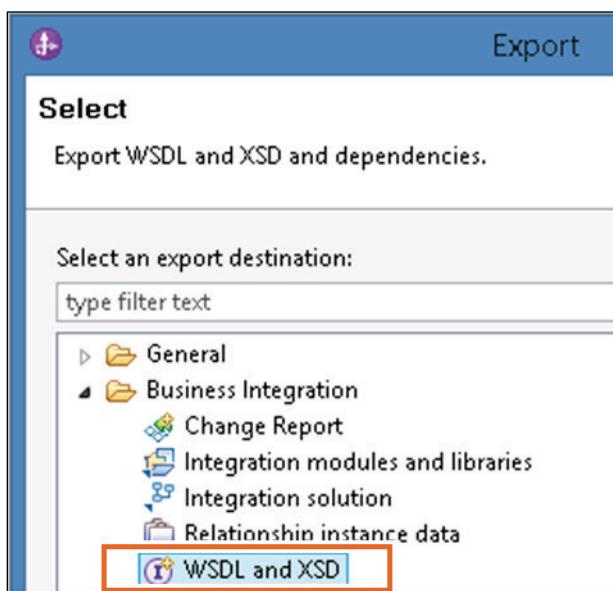
- Save the changes to the assembly diagram.
- Rename the web service export.
 - On the assembly diagram, select **CreditScoreServiceExport1** and switch to the **Properties** view.
 - With the cursor in the **Name** field, press Alt+Shift+R.
 - In the Rename Artifact dialog box, change the value in the **New name** field to: **CreditScoreServiceExport**



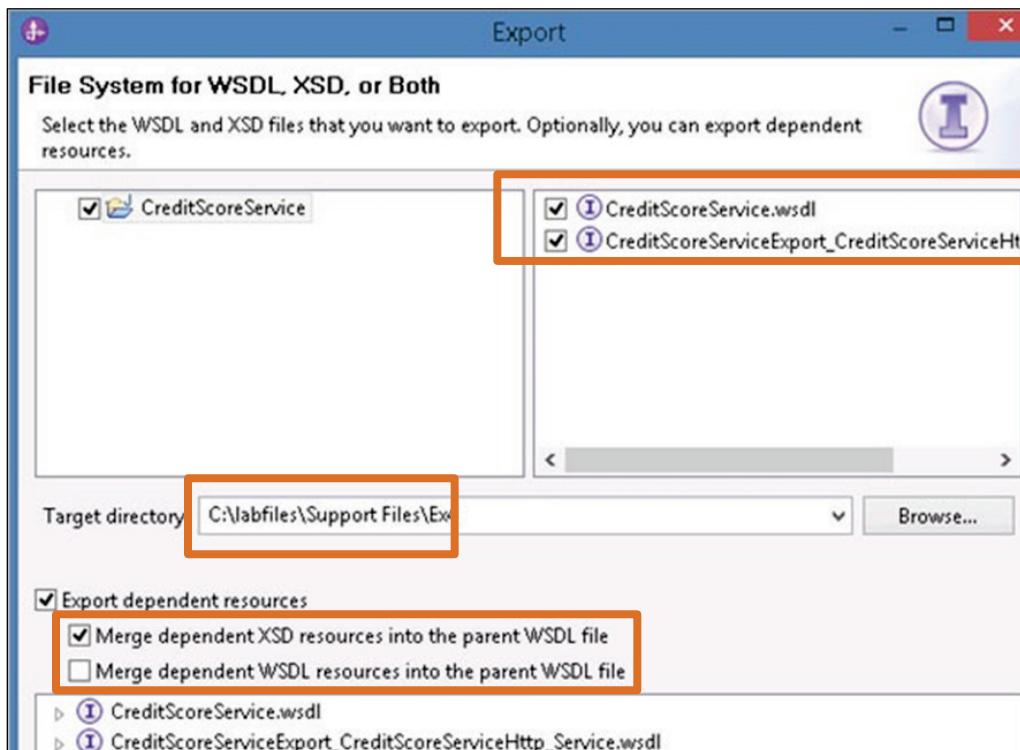
4. Accept the remaining default options and click **Refactor**.
4. Export the credit score service WSDL file and web service port definition.

You import them into a library that is named `FoundationLibrary`.

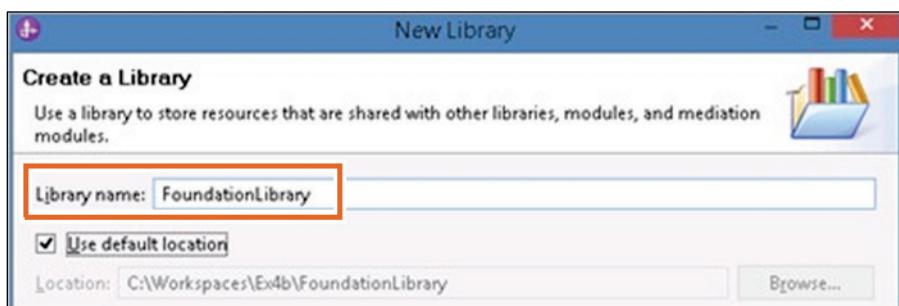
1. In the Business Integration view, right-click **CreditScoreService** and click **Export** from the menu.
2. In the Export dialog box, expand **Business Integration**, and select **WSDL and XSD**.
3. Click **Next**.



4. At the “File System for WSDL, XSD, or Both” window, do the following steps:
- Select **CreditScoreService.wsdl**, **CreditCheckRequest.xsd**, and **CreditScoreServiceExport_CreditScoreServiceHttp_Service.wsdl**.
 - For **Target directory**, click **Browse**. Browse to **C:\labfiles\Support Files\Ex3** and click **OK**.
 - Select **Merge dependent XSD resources into the parent WSDL file** (this action removes **CreditCheckRequest.xsd** from the window).



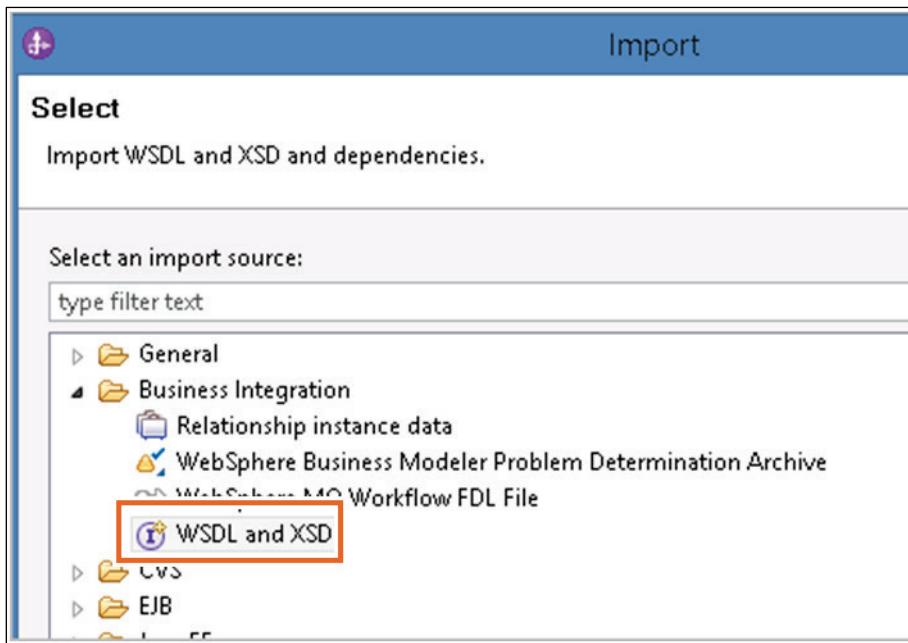
5. Accept the remaining default options and click **Finish**.
6. If you are prompted to overwrite any existing files, click **Yes**.
You will create a library that is named `FoundationLibrary` to store the WSDL interface and port definition.
7. Click **File > New > Library** from the menu options.
8. Type `FoundationLibrary` in the **Library name** field.



- Accept the remaining default options and click **Finish**.

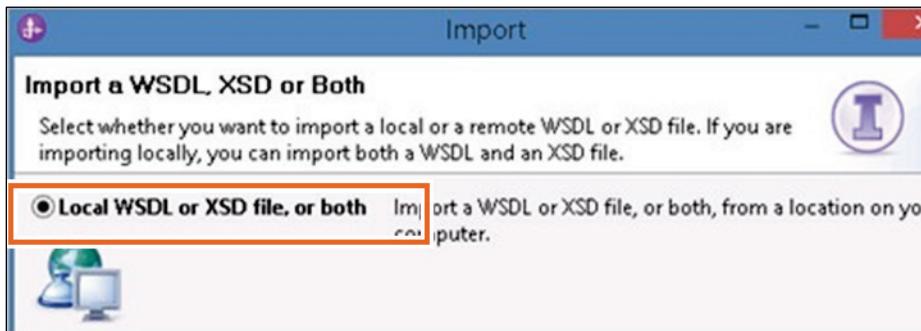
You will import the credit score service WSDL interface and port definition into FoundationLibrary.

- In the Business Integration view, right-click **FoundationLibrary** and click **Import** from the menu.
- Expand **Business Integration** and select **WSDL and XSD**.



- Click **Next**.

- In the "Import a WSDL, XSD or Both" dialog box, select **Local WSDL or XSD file, or both**.

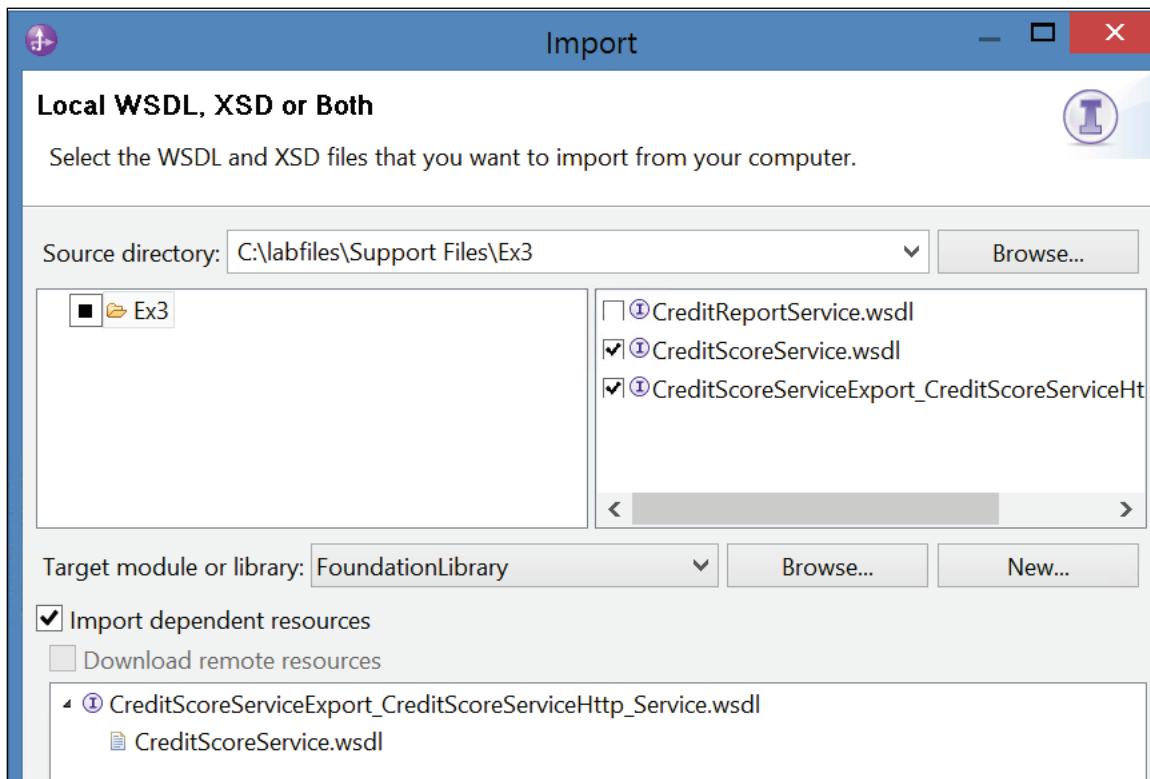


Note: You can also use the import wizard to import files from an external location through a URL.

- Click **Next**.

15. In the “Import a WSDL, XSD or Both” pane, take the following actions:

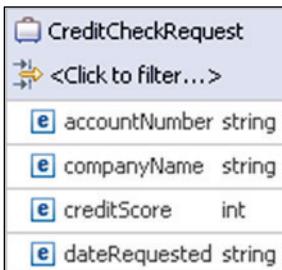
- For **Source directory**, click **Browse**, browse to C:\labfiles\Support Files\Ex3, and click **OK**.
- Select the **CreditScoreService.wsdl** and **CreditScoreServiceExport_CreditScoreServiceHttp_Service.wsdl** check boxes. Mandy Yao • Century 21 Abrams, Hutchinson & Associate
- Verify that **Target module or library** is set to **FoundationLibrary**.
- Verify that the **Import dependent resources** check box is selected.



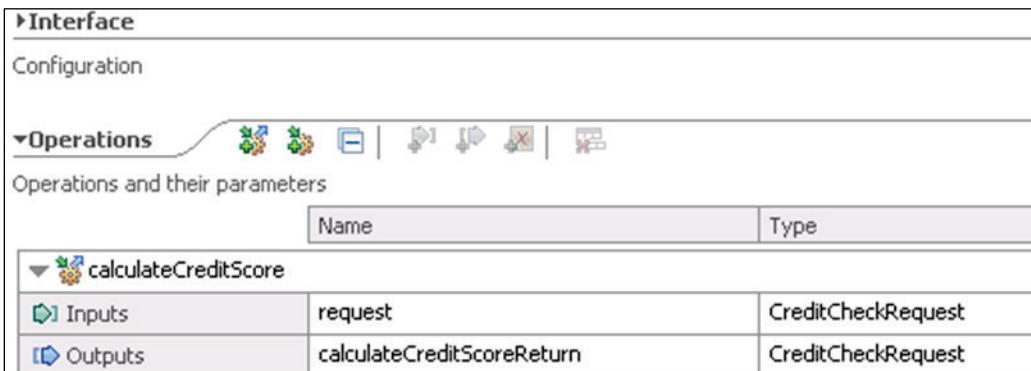
16. Accept the remaining default options and click **Finish**.

5. Examine the SCA artifacts in FoundationLibrary that are created as a result of the WSDL import.

1. Expand **FoundationLibrary > Data** and double-click **CreditCheckRequest** to open the business object definition.



2. Expand **Interfaces** and double-click **CreditScoreService** to open the interface definition.

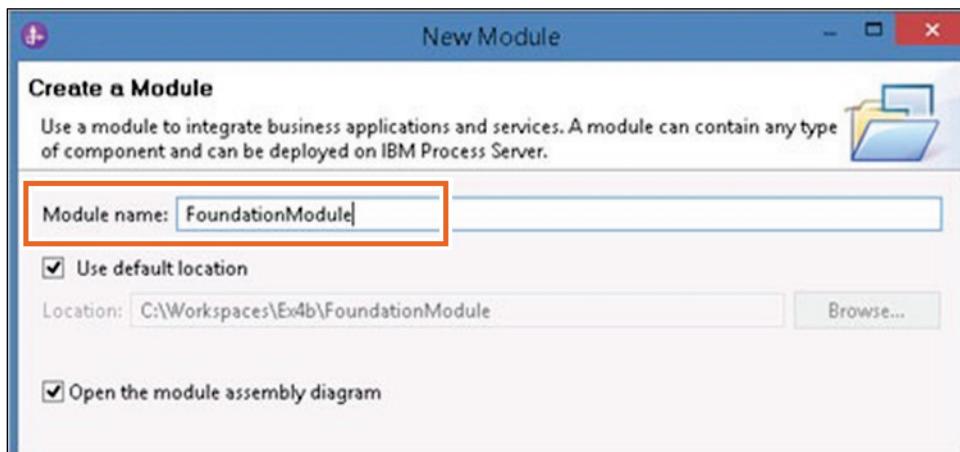


3. Close both the **CreditCheckRequest** tab and the **CreditScoreService** tab.

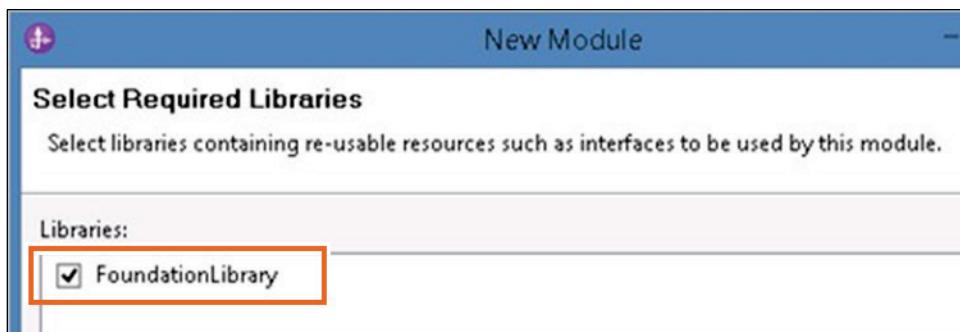
6. Create an IBM Integration Designer project that is named FoundationModule that includes FoundationLibrary in its dependencies.

This module is the starting point for the end-to-end application to develop throughout the remaining exercises.

1. Click **File > New > Module** from the menu options.
2. In the “Create a Module” dialog box, type **FoundationModule** in the **Module name** field.



3. Click **Next**.
4. In the Select Required Libraries dialog box, verify that **FoundationLibrary** is selected.



5. Click **Finish**.
6. Wait until no messages appear in the IBM Integration Designer status bar, such as Building workspace.
7. View the FoundationModule dependencies.
 1. Expand FoundationModule and double-click Dependencies.
 2. Expand the **Libraries** section.
 3. Verify that the **FoundationLibrary** is in the list.

Libraries	
Configure the required libraries. If a version is declared for a library, it must be an exact version. More...	
Name	Required Version
FoundationLibrary	

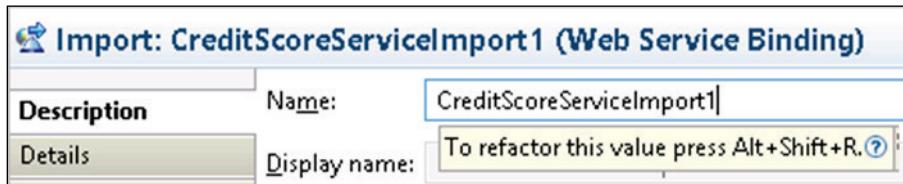
4. Close the **Dependencies** tab.

8. Add an import component that is named `CreditScoreServiceImport` to the `FoundationModule` assembly diagram that calls the credit score web service.

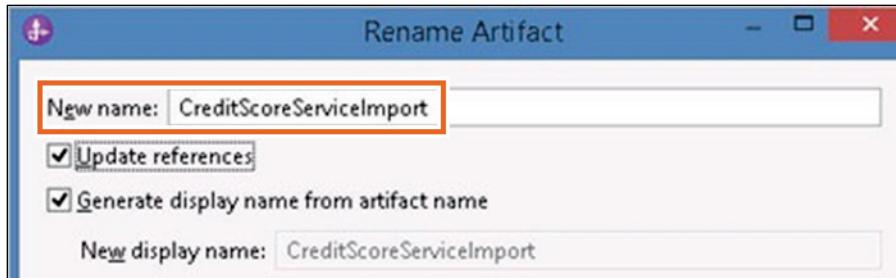
1. If the `FoundationModule` assembly diagram is not open, expand **FoundationModule** and double-click **Assembly Diagram**.
2. In the Business Integration view, expand **FoundationLibrary > Web Service Ports**.
3. Drag `CreditScoreServiceExport_CreditScoreServiceHttpPort` from **FoundationLibrary** onto the **FoundationModule** assembly diagram.
4. Save the changes to the assembly diagram.

9. Refactor the name of the import from `CreditScoreServiceImport1` to `CreditScoreServiceImport`.

1. With `CreditScoreServiceImport1` selected, switch to the **Description** tab in the **Properties** view.
2. With the cursor in the **Name** field, press **Alt+Shift+R**.



3. In the Rename Artifact dialog box, change the value in the **New name** field to: `CreditScoreServiceImport`



4. Accept the remaining default options and click **Refactor**.

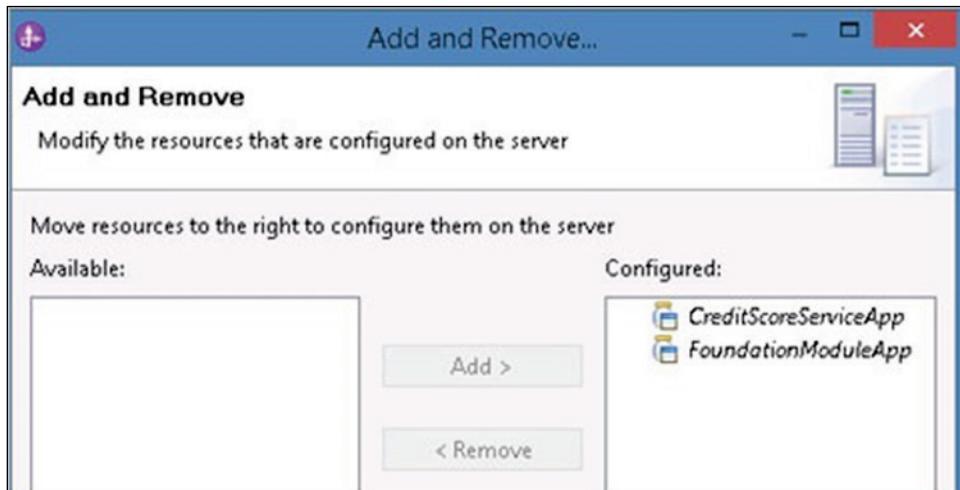
Part 7. Testing the web service import.

In this portion of the exercise, you test the web service import component. Using the IBM Integration Designer test client, you call the web service that is running in the local IBM Process Server test environment.

1. Test the credit score service import.

You are ready to deploy FoundationModule and CreditScoreService to the server and test the CreditScoreServiceImport component.

1. Start the process server if it is not running. You can do that by double-clicking the **Start UTE Process Server** shortcut.
2. Right-click **IBM Process Server v8.6 at localhost** and click **Add and Remove**.
3. Click **Add All** to add **FoundationModuleApp** and **CreditScoreServiceApp** to the Configured projects list.

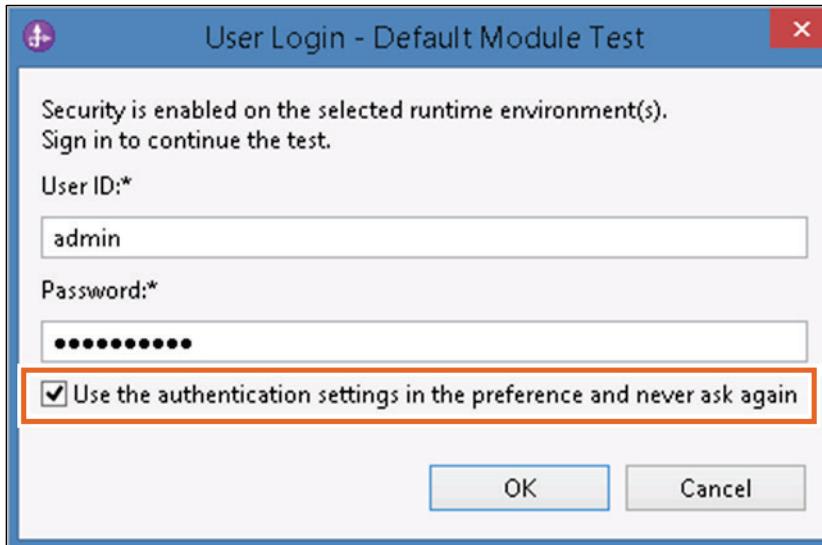


4. Click **Finish**.
5. Wait until the module is published and started.
6. In the assembly diagram, right-click **CreditScoreServiceImport** and click **Test Component** from the menu.
7. In the **Initial request parameters** table, enter the following test data:
 - acctNumber: 100
 - companyName: IBM
 - creditScore: 0
 - dateRequested: 04/15/2016

Name	Type	
request	CreditCheckDetail	[ab]
acctNumber *	string	[ab] 100
companyName *	string	[ab] IBM
creditScore *	int	[ab] 0
dateRequested *	string	[ab] 04/15/2016

8. Click **Continue** on the Events toolbar.

9. When the “Select a Deployment Location” dialog box is displayed, select **IBM Process Server v8.6 at localhost**, select **Use this location as the default and do not ask again**, and click **Finish**.
10. When the User Login dialog box is displayed, select the **Use the authentication settings in the preference and never ask again** check box, and click **OK**.



When the test completes, a stop node labeled “Invoke returned” is displayed in the Events window.

11. Select the **Return** event and examine the **Return parameters** section. The result of the test is a credit score of 11.

The Events window displays the following sequence of events:

- Invoke (CreditScoreServiceImport:calculateCreditScore)
- Invoke started
- Invoke (CreditScoreServiceImport:calculateCreditScore)
- Return (CreditScoreServiceImport:calculateCreditScore)
- Invoke returned

The Return parameters table shows the following data:

Name	Type	Value
calcul...	CreditCheck...	001
acco	string	IBM
corn	string	ACME
cred	int	11
date	string	04/15/2016

12. If time permits, you can test the additional test cases. If you enter the company name **AbcCo** or **TestCo**, the credit score that is returned is 1. If you enter the company name **ACME**, the credit score that is returned is 6.

Note: To rerun a test, click the **Invoke** button on the toolbar.



13. Close the test client tab. When you are prompted to save the test trace, click **No**.

2. Remove the projects.

14. In the **Servers** view, right-click **IBM Process Server v8.6 at localhost** and click **Add and Remove** from the menu.
15. Click **Remove All** and click **Finish**. Wait until no messages appear in the IBM Integration Designer status bar.
16. Close IBM Integration Designer.

Results

In this exercise, you imported an external WSDL file into IBM Integration Designer. You tested the connectivity with the web service and implemented SCA components in the IBM Integration Designer test environment. Finally, you used a web service export to expose an existing IBM Process Server application.

References

- OSOA: Service Component Architecture family of specifications
 - <http://oasis-open.org/sca>
- developerWorks: Building SOA solutions with the Service Component Architecture
 - Part 1
http://www.ibm.com/developerworks/websphere/techjournal/0510_brent/0510_brent.html
 - Part 2
http://www.ibm.com/developerworks/websphere/techjournal/0512_barclay/0512_barclay.html
 - Part 3
http://www.ibm.com/developerworks/websphere/techjournal/0602_barclay/0602_barclay.html
- IBM Redbooks: “Business Process Management Deployment Guide Using IBM Business Process Manager V8.5”
 - <http://www.redbooks.ibm.com/abstracts/sg248175.html?Open>

SCA bindings

© Copyright IBM Corporation 2018

References

Unit 7 Business objects

The slide features a blue header bar with 'IBM Training' on the left and the IBM logo on the right. The main content area has a light blue diagonal striped background. The title 'Business objects' is centered in large blue text. Below it, the text 'IBM Business Process Manager V8.6' is displayed. At the bottom, a copyright notice reads: '© Copyright IBM Corporation 2018' and 'Course materials may not be reproduced in whole or in part without the written permission of IBM.'

IBM Training

Business objects

IBM Business Process Manager V8.6

© Copyright IBM Corporation 2018
Course materials may not be reproduced in whole or in part without the written permission of IBM.

Unit objectives

- Identify the purpose and advantages of using the SDO framework
- Define the data object and data graph components of the SDO framework
- Identify the purpose and advantages of using the business object framework
- Define the business object and business graph components of the business object framework

Business objects

© Copyright IBM Corporation 2018

Unit objectives

This unit explains the Service Data Objects (SDO) architecture and framework, and how SDO is used in a solution.

Topics

- Introduction to Service Data Objects
- Introduction to business objects

Business objects

© Copyright IBM Corporation 2018

Topics

Introduction to Service Data Objects

Business objects

© Copyright IBM Corporation 2018

Introduction to Service Data Objects

Introduction to Service Data Objects (SDO)

- SDO provides a framework for data manipulation
 - SDO API unifies representation of data from multiple sources
 - Not necessary to know multiple technology-specific APIs
- SDO framework supports a disconnected programming model (manipulate data without a connection to the source)
- SDO is integrated with XML
- The key components of the SDO framework are data objects, data graphs, and data object metadata

Component	Description
Data object	Fundamental data structure for representing business data
Data graph	A container for a hierarchical set (a tree) of data objects
Data object metadata	Metadata is the schema definition of the object; it contains information about the data in the data object (property types, relationships, and constraints)

Business objects

© Copyright IBM Corporation 2018

Introduction to Service Data Objects (SDO)

It is not uncommon for an enterprise application developer to know several data access technologies, such as JDBC, XML, JMS, web services, and enterprise information systems. Unfortunately, developers are then required to become experts in many different data access technologies.

The goal of SDO is to provide a programming model that unifies data representation across heterogeneous data sources, and to simplify application development for developers and tools providers.

SDO provides a common API that can be used regardless of the data store that is being accessed. This common way of representing data also makes SDO an ideal choice for data abstraction in a service-oriented architecture.

Additionally, built into the SDO architecture is support for some common programming patterns. SDO supports a disconnected programming model. Typically a client might be disconnected from a particular data access service (DAS) while working with a set of business data. However, when the client completes processing, and must apply changes to a data store by way of a DAS, a change summary provides the appropriate level of data concurrency control. This change summary information is built into the SDO programming model.

Another important design point to note is that SDO integrates well with XML. As a result, SDO naturally fits in with distributed service-oriented applications.

Finally, SDO is designed to support both dynamic and static data access APIs. The dynamic APIs are provided with the SDO object model and provide an interface that allows developers to access data even when the schema of the data is not known until run time. In contrast, the static data APIs are used when the data schema is known at development time, and the developer prefers to work with “strongly typed” data access APIs.

SDO is a programming model that IBM and BEA proposed jointly as JSR 235. Since that time, SDO architects have published and revised specifications, which are available at the following websites:

- <http://oasis-open.org/sca>
- <http://www.jcp.org>

Data objects

- Data objects are the fundamental structures for representing business data
- A data object holds data as a set of properties
- Each data object provides read and write access to properties through:
 - Getters and setters
 - XPath (XML Path Language)
- Properties can be:
 - Primitive data types (such as strings)
 - Commonly used data types (such as dates)
 - Multivalued fields (such as arrays)
 - Other data objects
- In memory, data objects are represented as instances of `commonj.sdo.DataObject`
 - Objects are serialized to XML

Business objects

© Copyright IBM Corporation 2018

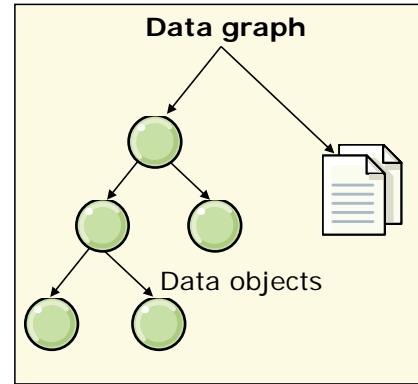
Data objects

The fundamental concept in the SDO architecture is the data object. It is not uncommon to see the term “SDO” used interchangeably with “data object.” A data object is a data structure that holds primitive data, multivalued fields (other data objects), or both. The data object also holds references to metadata that provides information about the data included in the data object. In the SDO programming model, the `commonj.sdo.DataObject` Java interface definition represents data objects. This interface includes method definitions that allow clients to get and set the properties that are associated with the **DataObject**. As an example, consider modeling customer data with an SDO data object. The properties that are associated with the customer might be: **firstName(String)**, **lastName(String)**, and **customerID(long)**. This sample shows how you would use the DataObject API to get and set properties for the customer data object:

```
DataObject customer = ...  
customer.setString("firstName", "John");  
customer.setString("lastName", "Doe");  
customer.setInt("customerID", 123);  
int id = customer.getInt("customerID");
```

Data graphs

- Data graphs are an optional wrapper around a root data object and associated data objects (in a tree structure)
 - Can include data objects from different data sources
- Data graphs include a change summary that records modifications to the data tree
 - Object changes: Reference to the object whose properties are changed, the properties that changed, the new value, and the old value
 - Object creations: Data objects that are added to the data graph
 - Object deletions: Data objects that are removed from the data graph



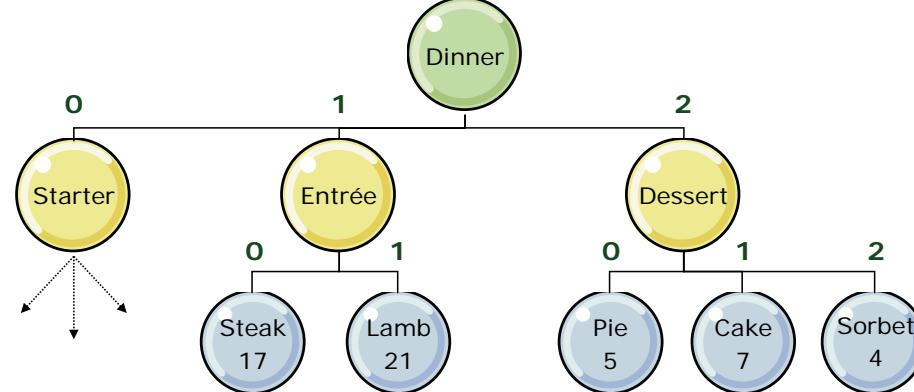
Data graphs

Another important concept in the SDO architecture is the data graph. A data graph is a structure that encapsulates a set of data objects. From the top-level data object contained in the graph, all other data objects are reachable by traversing the references from the root data object.

Another important feature included in the data graph is a change summary that is used to log information about the data objects in the graph that changed during processing. In the SDO programming model, the `commonj.sdo.DataGraph` Java interface definition represents data graphs. In addition, the `commonj.sdo.ChangeSummary` interface defines the change summary information. A complete object model for SDO V1.0 is included in the specification document.

Example: SDO data tree

- This data tree represents a dinner menu at a restaurant
 - The root object is a “menu type” with a value of Dinner
 - The related child objects are “course type” (Starter, Entrée, and Dessert) and “dish type” (Steak: \$17, and Lamb: \$21)
- Clients traverse the tree beginning at the root, and access the child objects by their index number: 0, 1, 2...n



Business objects

© Copyright IBM Corporation 2018

Example: SDO data tree

Introduction to business objects

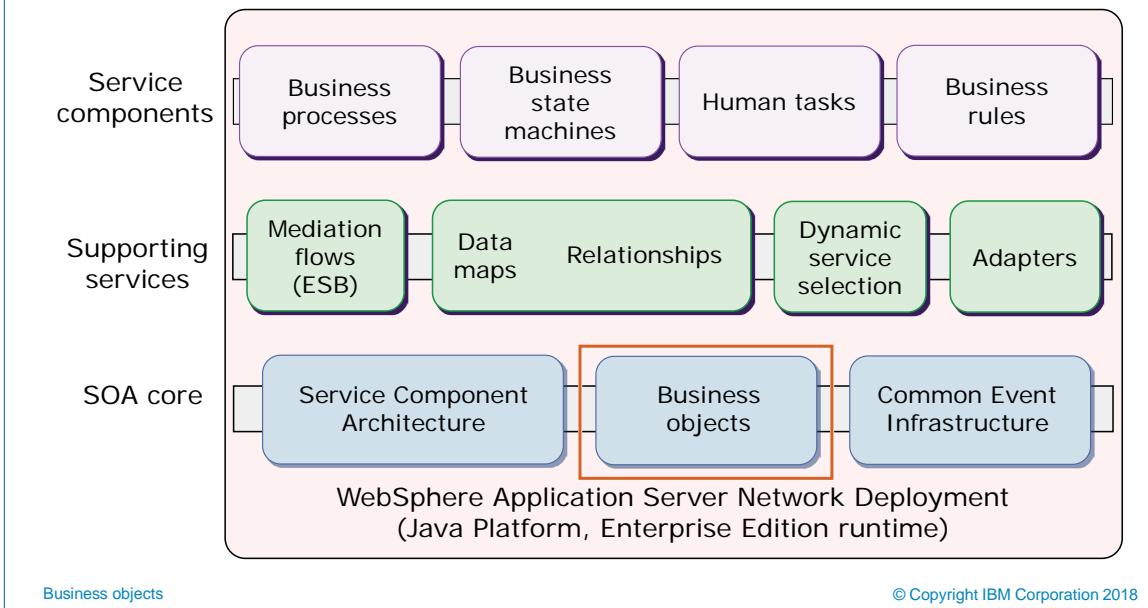
Business objects

© Copyright IBM Corporation 2018

Introduction to business objects

Business objects are an SOA core component

- Business objects are an SOA core component
- Business objects provide an abstraction layer for data objects



Business objects are an SOA core component

Business objects, which are extensions of Service Data Objects (SDO), provide an abstraction layer for data objects and are part of the SOA core components. Each of the SOA core components is based on an open standard: SCA, SDO, and Common Event Infrastructure. Business objects are a composite of the SDO specification and IBM extensions.

Introduction to the business object framework

- The business object framework is intended to provide a data abstraction for the Service Component Architecture (SCA)
 - The business object framework and common API are designed to mitigate complexities of working with disparate business data in SOA
 - The business object framework provides some additional business integration functions that are not found in SDO
- Business objects represent the data that flows between SCA components in a service-oriented architecture
- Component interfaces use business objects as inputs and outputs

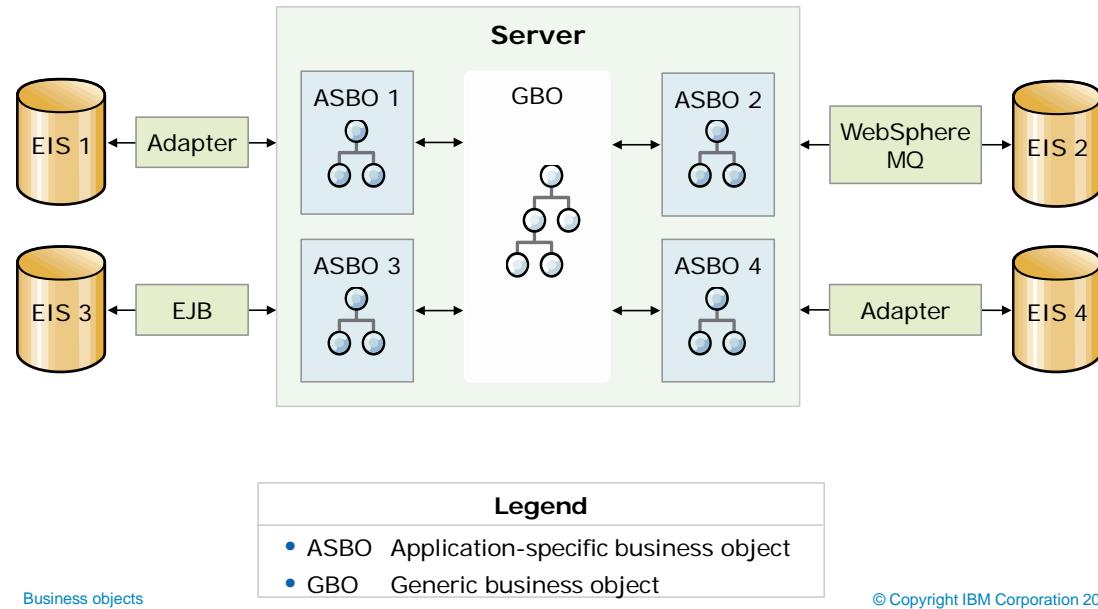
Introduction to the business object framework

Server products such as IBM Business Process Manager implement the SDO specification by using business objects. SCA components exchange data by passing business objects to each other. Business objects (SDOs) are thus the primary data abstraction for the Service Component Architecture (SCA).

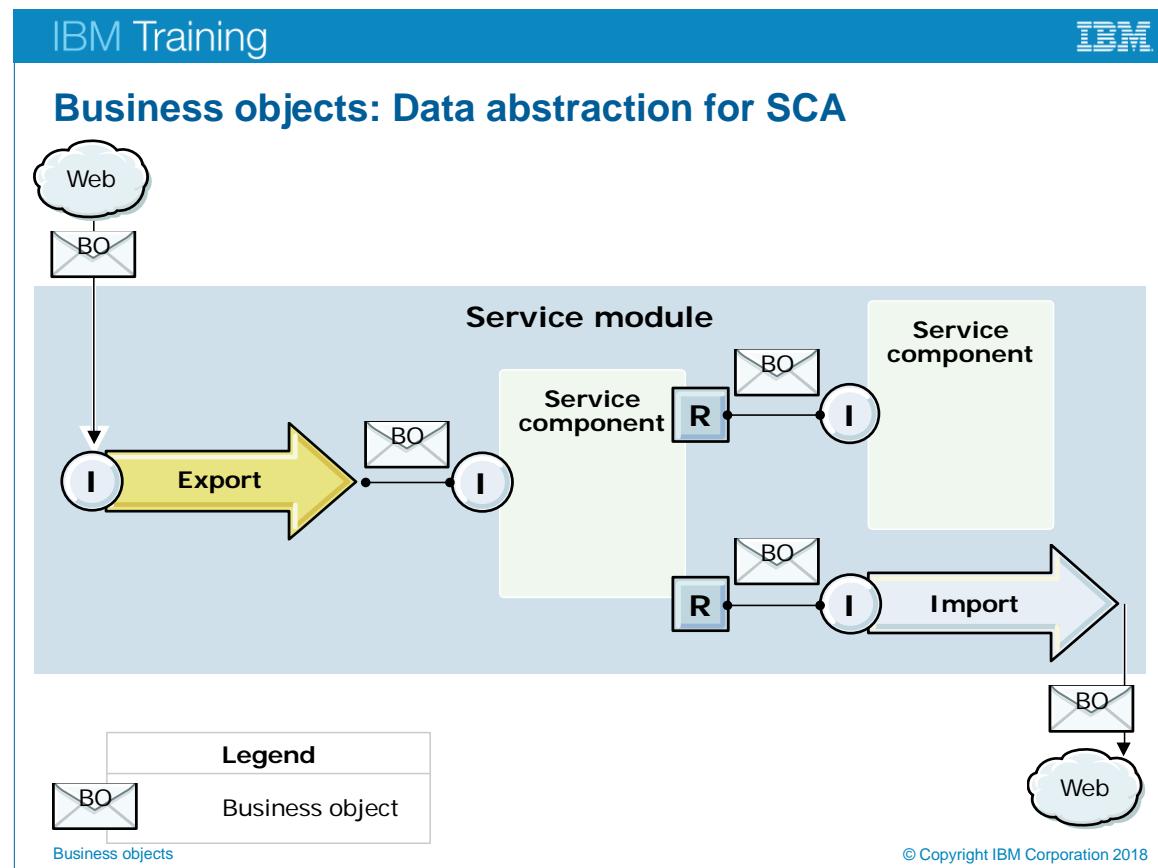
The business object framework in IBM Process Server is based on SDO technology. However, the SDO specification does not include business integration capabilities such as “verbs” or “event summaries.” WebSphere Adapters and earlier versions of IBM Process Server require these capabilities. The event summary and verb are now deprecated. The added features that are provided are not part of the core SDO specification.

Business objects that are represented as SDOs

- All business objects are implemented as SDOs



Business objects that are represented as SDOs



Business objects: Data abstraction for SCA

Business objects (SDOs) are the primary data abstraction for the Service Component Architecture (SCA).

This diagram illustrates how SCA provides the framework to define service components and compose these services into integrated applications. It further shows that business objects represent the data that flows between services. Whether the interface that is associated with a particular service component is defined as a Java interface or a WSDL portType, the input and output parameters are represented by using business objects. The data abstraction for SCA is achieved through business objects (SDO), and the SCA programming model is used to manage invocation.

Composition of business object framework

- The business object framework is composed of the following components:

Component	Description
Business object (BO)	Fundamental data structure for representing business data
Business graph (BG)	Optional wrapper for a business object or hierarchy of business objects to provide enhanced information such as: <ul style="list-style-type: none"> Change summary Event summary (IBM extension to SDO): Deprecated Verb (IBM extension to SDO): Deprecated
Business object type metadata	<ul style="list-style-type: none"> Annotate business objects with application-specific information
Business object services	<ul style="list-style-type: none"> A set of services that facilitate working with business objects These services are in addition to the capabilities provided in SDO

Business objects

© Copyright IBM Corporation 2018

Composition of business object framework

The table on this page introduces several key concepts that make up the business object framework. The intent is to provide a high-level picture of the major pieces that compose the entire framework before considering the details of each concept.

The business object is the fundamental data structure that is used to represent business data. It is not uncommon to use the term business object to refer to the entire framework. However, in this presentation when the term business object is used, it refers to the fundamental data structure for representing business data and not to the overall architecture.

The business graph is used in the business object framework to wrap a top-level business object and to provide more information that is used to enhance the data. In particular, the data graph includes the change summary for the data in the graph (similar to the SDO change summary information), event summary, and verb information (used for data synchronization between EIS systems). The business graph is similar to the concept of the data graph in the SDO architecture. However, notice that the event summary and the verb portion of the enhanced information are not included with the SDO data graph concept. The event summary and verb are described in more detail. The two extensions to the SDO framework are deprecated.

Verbs were previously used to communicate the type of after-image (enterprise information system) event that was being moved through the business integration runtime environment. A verb provided information to adapters about what type of action to take on a business object in the EIS to which it was connected. If the verb was “create”, for example, then the change summary would be empty (no change is needed). The most common verbs that adapters supported were:

- Create (create a record in the EIS)
- Retrieve (retrieve a record from the EIS)
- Update (update a record in the EIS)
- Delete (delete a record from the EIS)
- Update with delete (update one record in the EIS while deleting another related record)

However, the list of supported verbs can be restricted, or users can specify more verbs. Verbs can be extended beyond the standard set, or they can be constrained to a particular set of values.

Typically, adapter developers are the ones who used the event summary in previous releases. The event summary had two purposes:

- To carry the ObjectEventID
- To provide an extensible markup metadata mechanism for DataObjects at run time

ObjectEventID is the mechanism that is used to uniquely identify an instance of an object that is displayed in the runtime environment. In previous versions, this information was appended to each business object that was designed top-down. In the business object specification, this information must be separated from the original business object to remain non-intrusive. Thus, in the business object framework, this information was carried in the event summary, where the unique identifier is associated with a DataObject in the business object hierarchy of the business graph.

Event information can also be carried in the event summary. This summary is a string that can be used to add metadata that is associated with each object in the business object hierarchy of the business graph. One potentially useful model for event information in the event summary is to mark up contained business objects with a verb other than the standard create, update, and delete verbs. The event summary is also used to add metadata to business objects at run time. For example, you can use the event summary to carry a verb other than the standard verbs.

The business object type metadata is available to annotate business objects with application-specific information.

Business object services are a set of services that facilitate working with business objects. These services are available in addition to the capabilities that are already provided by SDO and are needed to provide function where the initial version of SDO did not specify a solution. In the revised SDO specification, these types of services are included where applicable. The business object services are a set of business object service APIs provided to enhance existing SDO capabilities, including business object create, copy, compare, and serialize. They also include services to access enhanced business graph capabilities: event summary and change summary.

Business objects (1 of 2)

- Business objects are the primary data structure for business data and data types that are defined in WSDL (interface) definitions
- Business objects are modeled by using XML schemas (XSD)
 - Can import business object schema definitions from, or export to, other systems
 - Support for the full XML schema data type system
- At run time, `commonj.sdo.DataObject` is used to represent business objects in memory as an SDO instance
 - Created from XSD files by using the business object factory
 - Accessible by using the SDO API and XPath
- Support is provided for data object schemas from industry standards organizations
 - HL7, ACORD, and OAGIS
 - IBM Business Process Manager Industry Packs provide prebuilt data objects

Business objects

© Copyright IBM Corporation 2018

Business objects

A business object is the primary structure for representing business data in the IBM Process Server runtime environment. This structure also includes document literal message definitions as would be found in a WSDL definition when defining included data types. The business object relates to the DataObject construct in SDO. In fact, business objects are represented in memory as an SDO `commonj.sdo.DataObject`. Therefore, if you are doing development work that involves programmatically working with business objects, it is important to become familiar with the SDO APIs. Also, you might occasionally see a business object that is referred to as an SDO because the SDO DataObject is used to represent a business object in the client programming model.

Currently, the only model for modeling or defining business objects is XML schema (XSD). The business object framework supports the full XML schema type system and facet capabilities. For this reason, business object definitions that third-party systems create can be successfully imported and used in an IBM Process Server application module. Development teams that might use a model other than XML schema to define business objects must convert these business objects to XML schema for use in IBM Process Server.

IBM Integration Designer support for industry standard organizations extends beyond the ones that are mentioned here (HL7: Health Level Seven, ACORD: Association for Cooperative Operations Research and Development, OAGIS: Open Applications Group Integration Specification). For a complete list of supported industry schemas, consult the IBM Integration Designer product documentation.

Corresponding to the addition of support for industry standard schemas, IBM Integration Designer also gives you the ability to exclude libraries when cleaning a project. This feature is especially useful for artifacts that are stable, such as industry schemas. When you use the **Project > Clean** menu item to invoke a full build of projects in the workspace, the resources in all of the libraries are automatically revalidated. If you have libraries that contain large XML schemas or WSDL files, the process of revalidating the library resources can considerably add to the time required for the build to complete. However, if you have one or more libraries that contain large files and you do not expect the libraries to change, you can exclude them from the builds and then reduce the overall build time.

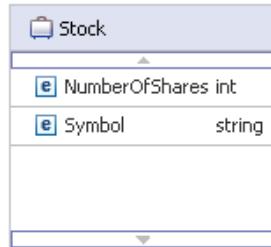
Working with industry standard schemas often means working with many business objects. In IBM Integration Designer, you can enter a filter in the parent category so that you can quickly search and find an artifact that you want. For example, if you have many business objects in the data types category, you can use inline filtering to quickly locate a specific business object. You can also mark artifacts as favorites, which do not get filtered, so you can tailor the Business Integration view to show only particular artifacts. These features are especially useful when working with industry schemas.

Business objects (2 of 2)

- Business objects are collections of elements with names and data types

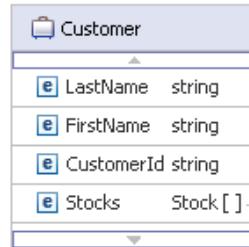
Two types of business objects:

1. Simple business objects that are composed of scalar (single-value) elements



Business objects

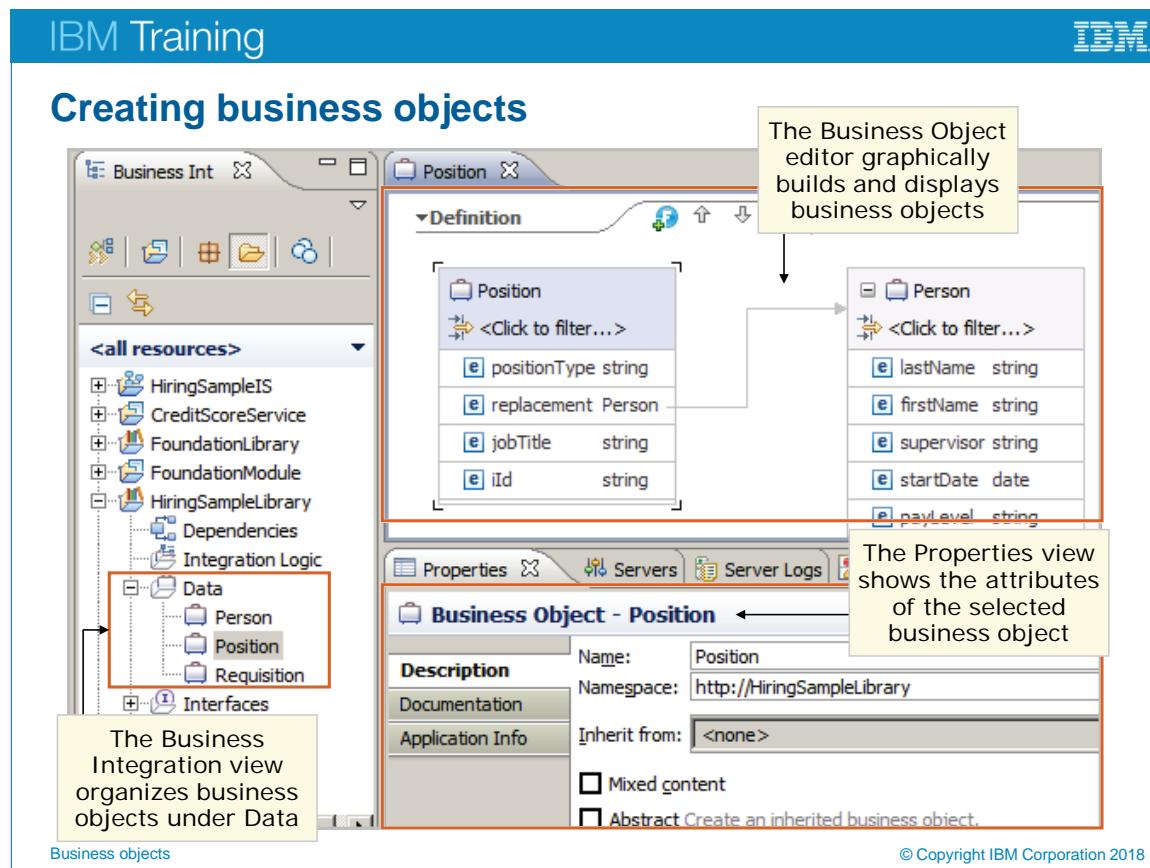
2. Hierarchical business objects with elements that contain other (child) business objects



© Copyright IBM Corporation 2018

Business objects have two types. In the simplest case, a business object can be constructed of only scalar properties as is shown in the Stock business object. In addition, business objects can be defined to be hierarchical (composed of one or more properties that reference a nested business object definition).

The Customer business object that is shown on this slide is included to illustrate an example of a hierarchical business object. In this example, each customer is associated with a collection of stocks. The data type for the Stocks property is the Stock business object, represented as an array.



Creating business objects

The Business Integration perspective has a Business Integration view, which provides a logical view of the key resources in each module and library. Within each module or library, the resources are categorized by type. Logical resources that are shown in the navigation tree do not necessarily have a one-to-one mapping to files. Artifacts that are not necessary for the development of integration applications are not shown in this navigation tree. Business objects are listed under the library data types section. When you open a resource from the Business Integration view with an editor, the resource is displayed in the editor area. The diagrams for the business object editor are composed on the canvas of the editor. When you use the Properties view with editors, you can modify properties of selected elements in the editor.

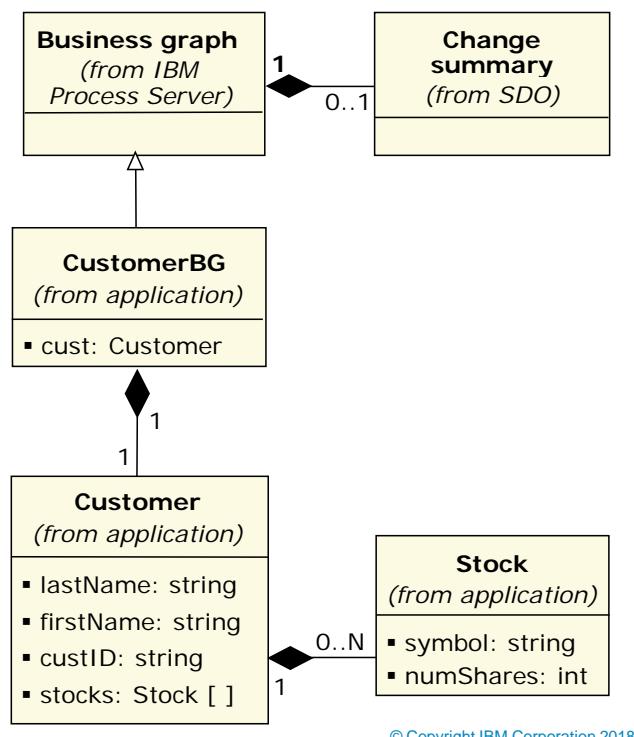
To define new business objects by using the business object editor, complete the following steps:

1. Expand the new module that is located inside the Business Integration section of the Integration Designer window.
2. Right-click the Data Types folder and select **New > Business Object**.
3. Type a new Name in the Business Object window. For example, type `Customer` to create a customer business object.

4. Click **Finish**. The new business object is added to the Data Types folder.
5. Click the “Add a field to a business object” icon and add the necessary fields to the business object.
6. Click the Save icon.
7. Repeat the previous steps for each business object that you want to create.

Business graphs

- A business graph is an optional container around a business object or a hierarchy of business objects
- Business graphs are SDO data graphs with extensions
- Business graphs include:
 - Root and associated business objects (in a tree)
 - A change summary header
- Graphs are seldom used outside the adapters (some adapters require graphs for the change summary)



Business objects

© Copyright IBM Corporation 2018

Business graphs

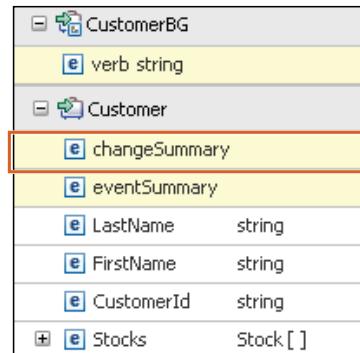
This slide shows a view of the programmatic design of business objects. The class diagram shows the relationship between SDOs and BOs, and the relationship between business graphs and business objects.

A business graph is a concept that is related to the SDO data graph concept. In the IBM Process Server runtime environment, a business graph is used as a container to wrap a top-level business object and provides several important capabilities that are related to the contained business object hierarchy.

Business graphs are available in the runtime to provide supporting metadata to adapters that are connected to back-end systems. Depending on the adapter, a business graph might not be required to indicate to the back-end system what this data represents, or what action should be taken regarding it. Business graphs are optional and are required only when you are adding business objects to a module created with an older version. If business graphs exist, they are processed, but the verb is ignored.

Business graph: Change summary

- The change summary is used to record changes to contained business objects
- Change summaries support disconnected data patterns and command-event models
 - Disconnected data pattern:** Records changes to business objects while disconnected from the data source
 - Command event model:** Allows an EIS to capture and publish data changes for use by other systems that must be aware of these events
- Two types of change summary usage:
 - Implicit:** Change logging is turned on, and any changes to contained business objects are logged
 - Explicit:** Modify the change summary explicitly (IBM extension)



Business graph: Change summary

The Customer business object in the figure needs the enhancements that a business graph provides. To produce these enhancements, you would create a business graph called **CustomerBG**. The **CustomerBG** is defined by using XML schema and can be thought of as a specialized type of business object that includes the verb for the business graph and a single business object of the Customer.

Change summaries support two use cases. The first use case is a disconnected use model where it is necessary to be able to capture changes that are made to the contained business object between processes. For example, consider a scenario where a service (service1) constructs a business graph that is based on a query to a data store and then passes that business graph to another service (service2) for processing. The second service can change the business objects that are contained in the business graph and then later call the first service to apply those changes to the data store. In this case, change summary information is necessary to assure the appropriate level of data concurrency.

Another important use case is the situation where the information that is contained with the business graph is used to capture and publish data changes to the business integration runtime environment from an EIS system. In this case, the change summary, event summary, and verb that is associated with the business graph are all

used to communicate the appropriate information that is needed for data synchronization among different EIS systems.

Some more notes on business graphs are as follows:

- Business graphs are used for enrichment of only top-level business objects. However, a contained business object in one scenario can still be a top-level business object in another scenario and a candidate for enrichment with a business graph.
- A business graph is represented as a `commonj.sdo.DataObject` in memory, but the event header and the change summary header cannot be accessed through the normal SDO APIs that are used to access contained DataObject properties.
- The change summary that is associated with a business graph can be used in two ways. The first way is to make implicit change summary updates. These updates are made by turning on change summary logging through the `commonj.sdo.ChangeSummary` interface. Doing so begins tracking changes that are made whenever any of the DataObject APIs are called for the business objects that are contained in the business graph.
- The other type of change summary tracking can be made explicitly by using the business object service that allows the change summary to be updated directly. This service must be offered as a business object service because this capability is not available with the base SDO change summary function, and it is needed to support the command-event model. An example of explicit change summary usage is in the course appendixes.

Data objects in IBM Business Process Manager

IBM Integration Designer

- Supports modeling and developing integrated data objects
- Business objects follow the SDO framework
- Business objects are modeled by using the XSD standard
- Runtime objects are instances of `commonj.sdo.DataObject`

IBM Process Designer

- Supports modeling data objects
- SDO objects are not supported
- Business graphs are not supported, so:
 - No change summaries exist
 - Disconnected data model is not supported
 - Command event model is not supported
- Data objects are stored as XML BLOBs in the IBM Process Center repository
- Data objects become instances of JavaScript objects at run time
- Data objects might be exported as XML for use in IBM Integration Designer

Data objects in IBM Business Process Manager

Process Designer and Integration Designer provide developers with one common business object model for representing different kinds of business entities from different domains.

In Process Designer, business objects are focused on a data type representation. Base business objects (variable types) are provided in system toolkits, and you can create custom variable types called custom business objects. In Integration Designer, objects can represent more complex XSD constructs. In Integration Designer, business objects have a close affinity with XML schemas.

At development time in Integration Designer, the business object model enables developers to define business objects as XML schema definitions. At run time, the business data that is defined in the XML schema definitions is represented as Java business objects. In this model, business objects are loosely based on early drafts of the Service Data Object (SDO) specification and provide the complete set of programming model application interfaces required to manipulate business data.

Unit summary

- Identify the purpose and advantages of using the SDO framework
- Define the data object and data graph components of the SDO framework
- Identify the purpose and advantages of using the business object framework
- Define the business object and business graph components of the business object framework

Checkpoint questions

1. What is the purpose of the business object framework?
2. Name two ways in which the business object framework differs from the Service Data Object framework.
3. What is the purpose of a change summary?
4. How are business objects represented in memory at run time?
5. What is the purpose of the “business object compare” utility?

Checkpoint answers

1. Service Data Objects provide a framework for application data and unify the representation of data from multiple sources by using a single API for data access: the SDO API.
2. The business object framework supports event summaries, verbs, and explicit manipulation of the change summary. The SDO framework does not.
3. The change summary is used to record changes to contained business objects.
4. At run time, `commonj.sdo.DataObject` is used to represent business objects in memory as an SDO instance.
5. It shows the structure of business objects in two different modules or libraries, and you can visually determine additions, removals, and changes. Also, with the “copy report to clipboard” option, you can paste changes into a spreadsheet, document, or other report.

Supplemental note: Business object framework runtime environment

- The business object framework runtime is:
 - Built on a high-speed XML infrastructure that enables optimized XML processing and enhanced XML fidelity
 - An internal engine that is visible to process developers by using the business object, XPath, and XSLT programming models available in the IBM Process Server SCA components
- This table describes how data in the runtime is modeled and integrated:

Abstraction	Implementation	Description
Instance data	Business object	Primary mechanism for representing business objects
Instance metadata	Business graph	Wrappers around simple business objects that carry a change summary
Type metadata	Enterprise or business object type metadata	Might be added to business object definitions to enhance their runtime value
Services	Business object services (APIs)	Set of capabilities that are provided on top of the basic SDO capabilities

Business objects

© Copyright IBM Corporation 2018

Supplemental note: Business object framework runtime environment

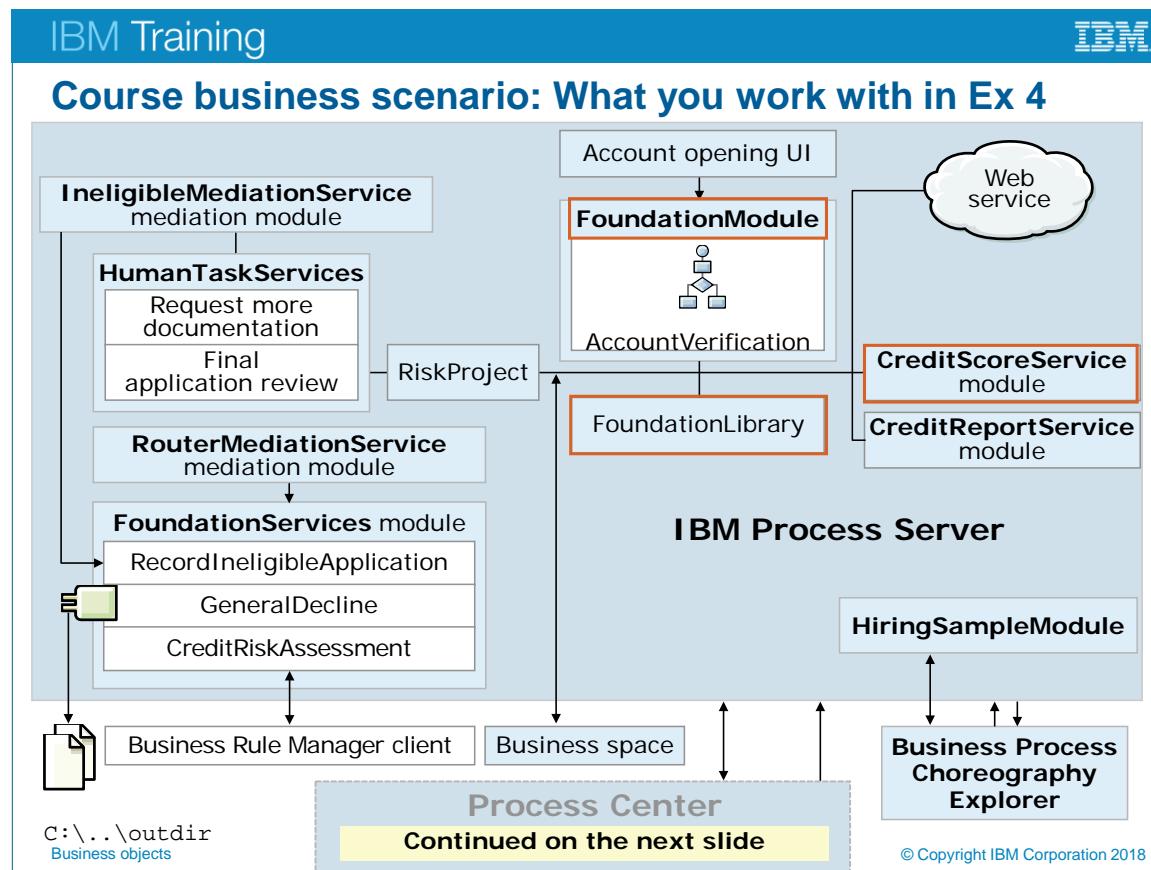
Exercise 4: Creating business objects and shared interfaces

After completing this exercise, you should be able to:

- Implement business objects and define their elements
- Implement interfaces that use business objects as inputs and outputs

Exercise 4: Creating business objects and shared interfaces

In this exercise, you create business objects and interfaces that the components use in your end-to-end solution. These artifacts are created in a library for use by multiple modules.

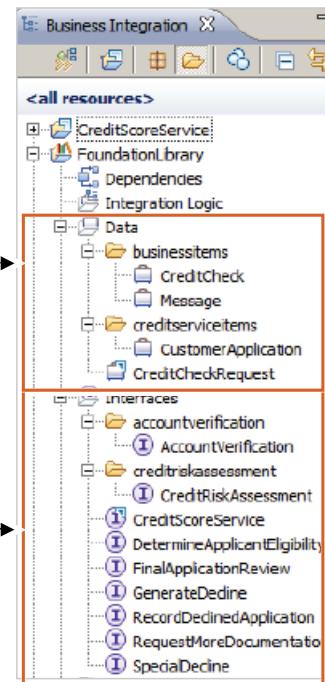


Course business scenario: What you work with in Ex 4

Components that are required for Exercise 4 (1 of 2)

Prebuilt components that are imported in the lab:

1. **FoundationModule** module
2. **CreditScoreService** module
3. **FoundationLibrary** container for **business objects**:
 - CreditCheck
 - Message
 - CustomerApplication
 - CreditCheckRequest
4. **FoundationLibrary** container for **interfaces**:
 - AccountVerification
 - CreditRiskAssessment
 - CreditScoreService
 - DetermineApplicantEligibility
 - FinalApplicationReview
 - GenerateDecline
 - RecordDeclineApplication
 - RequestMoreDocumentation
 - SpecialDecline



Business objects

© Copyright IBM Corporation 2018

Components that are required for Exercise 4

Components that are required for Exercise 4 (2 of 2)

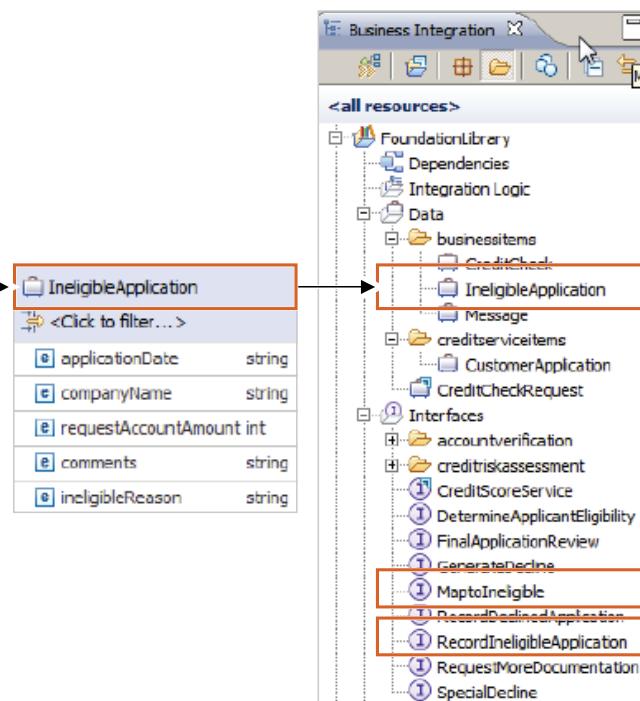
New components that you create in the lab:

1. Business object:

- IneligibleApplication

2. Interfaces:

- MptoIneligible
- RecordIneligibleApplication



Exercise 4: Creating business objects and shared interfaces

Purpose:

In this exercise, you implement business objects with multiple attributes in the library you created previously. You also define abstract interfaces in the library that have request and response operations and use the defined business objects as inputs and outputs.

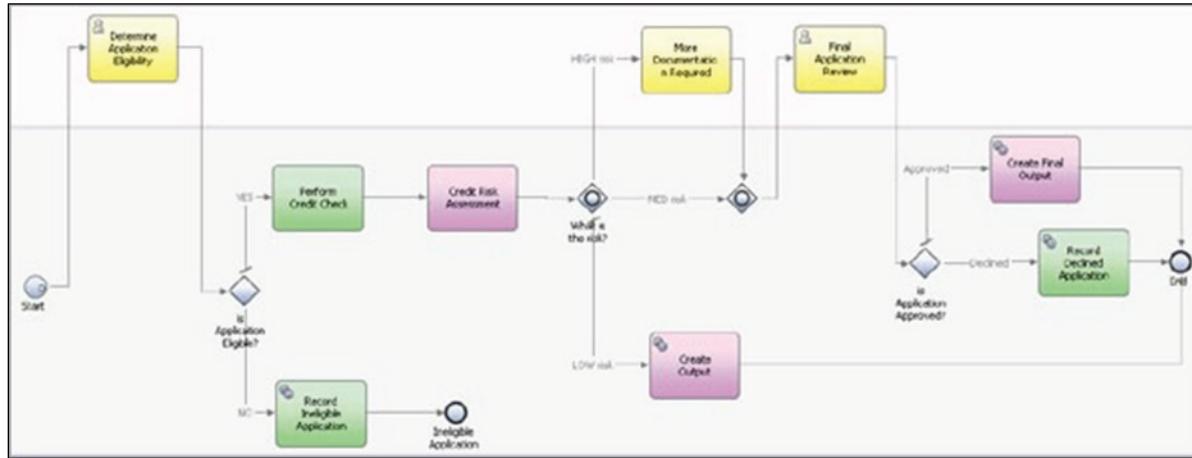
Completing the exercises for this course requires a lab environment. This environment includes the exercise support files, IBM Process Designer, IBM Process Center, and IBM Integration Designer test environment.

Exercise Instructions

In this exercise, you use the business object editor in IBM Integration Designer to define a business object. You also use the interface editor to implement the RecordIneligibleApplication and MaptoIneligible interface definitions. Because multiple modules can use business object definitions and interface definitions, you implement these definitions in a library, FoundationLibrary.

The account verification scenario uses the business object. The business objects in this exercise are used at different levels of the business process.

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.



1. **CustomerApplication.** This business object is the most widely used of the process application. A new application is submitted, and it is the goal of the process to handle that customer application. This business object is created for you.
2. **IneligibleApplication.** This business object is used if the application is not eligible (as determined by the **DetermineApplicationEligibility** activity, which sets the *eligibleApplication* field of the **CustomerApplication** business object). You create this business object.
3. **CreditCheckRequest.** In a previous exercise, you created a service component, which called a web service. The purpose of that web service was to check the customer's credit. That web service requires a specific business object. You might recall that **CreditCheckRequest** was imported with the web service definition in a previous exercise.
4. **Message.** The response of the business process application is to return this Message business object. This business object is created for you.

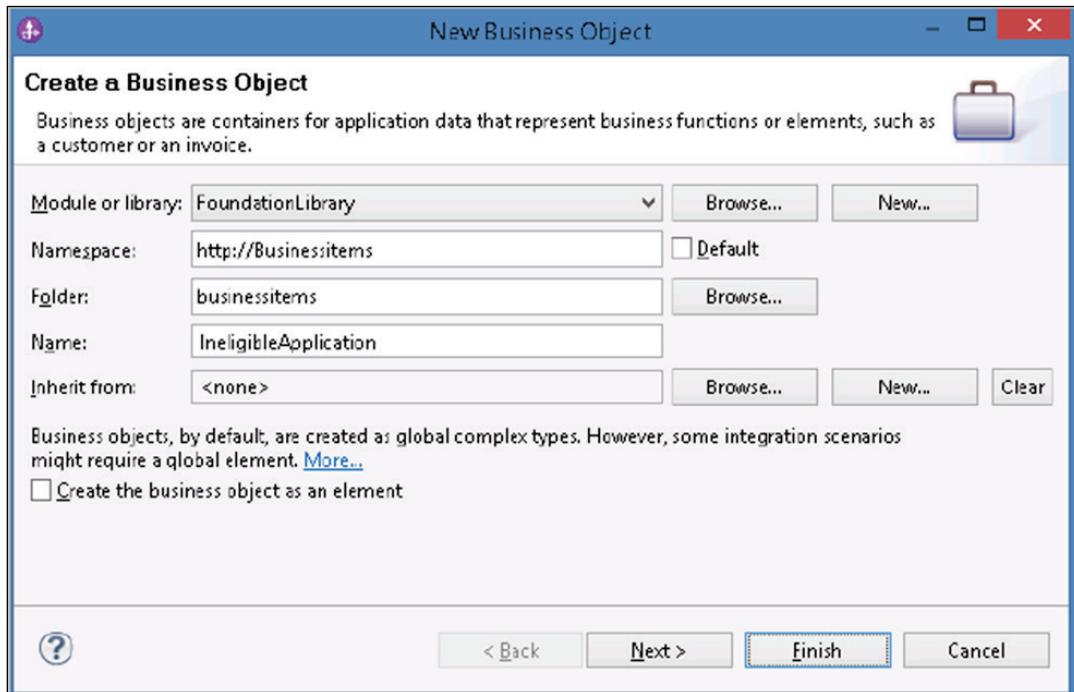
Part 1. Implement business objects and define their elements.

In this portion of the exercise, you create the **IneligibleApplication** business object that is required for the account verification business process application.

1. Create the business object.
 1. On your desktop, open the folder that is labeled **Exercise Shortcuts**.
 2. Double-click the shortcut that is labeled **Exercise 4**. Allow Integration Designer a few moments to build the workspace. You can view the workspace build status at the lower-right corner of the Integration Designer. Wait until the status reaches 100%, at which point the workspace is built, and the status progress bar disappears. If you get a message that the server is already set to publish, then click **OK**.
 3. Close the **Getting Started** tab.
2. Use the `http://Businessitems` namespace to create a business object, **IneligibleApplication**, in the `businessitems` folder.
 1. Expand **FoundationLibrary**, right-click **Data**, and click **New > Business Object** from the menu.
 2. At the "Create a Business Object" dialog box, take the following actions:
 - Verify that the **Module or library** field is set to `FoundationLibrary`.
 - Clear the **Default** check box, and change the **Namespace** field to:

`http://Businessitems` (note the case).

- Type `businessitems` in the **Folder** field (note the case), or click **Browse** to select it.
- Type `IneligibleApplication` in the **Name** field.



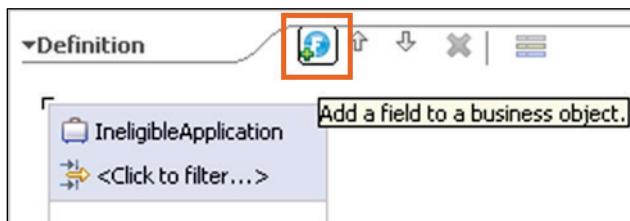
3. Accept the remaining default options and click **Finish**.

The **IneligibleApplication** business object opens in the business object editor.

3. Add the following fields to the **IneligibleApplication** business object:

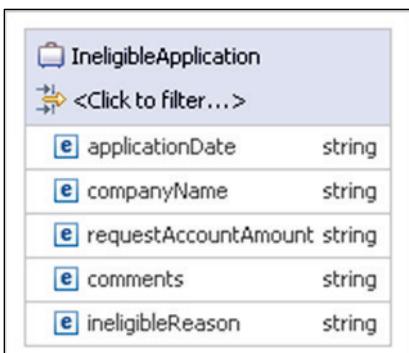
- `applicationDate(string)`
- `companyName(string)`
- `requestAccountAmount(int)`
- `comments(string)`
- `ineligibleReason(string)`

1. In the business object editor, with **IneligibleApplication** selected, click the **Add a field to a business object** icon.



2. Change the default field name `field1` to: `applicationDate`
3. With **IneligibleApplication** selected, click the **Add a field to a business object** icon.
4. Change the default field name `field1` to: `companyName`
5. With **IneligibleApplication** selected, click the **Add a field to a business object** icon.
6. Change the default field name `field1` to: `requestAccountAmount`
7. With **IneligibleApplication** selected, click the **Add a field to a business object** icon.
8. Change the default field name `field1` to: `comments`
9. With **IneligibleApplication** selected, click the **Add a field to a business object** icon.
10. Change the default field name `field1` to: `ineligibleReason`

The completed business object resembles the following figure:



However, as indicated in a previous step, the `requestAccountAmount` type is expected to be an integer, not a string.

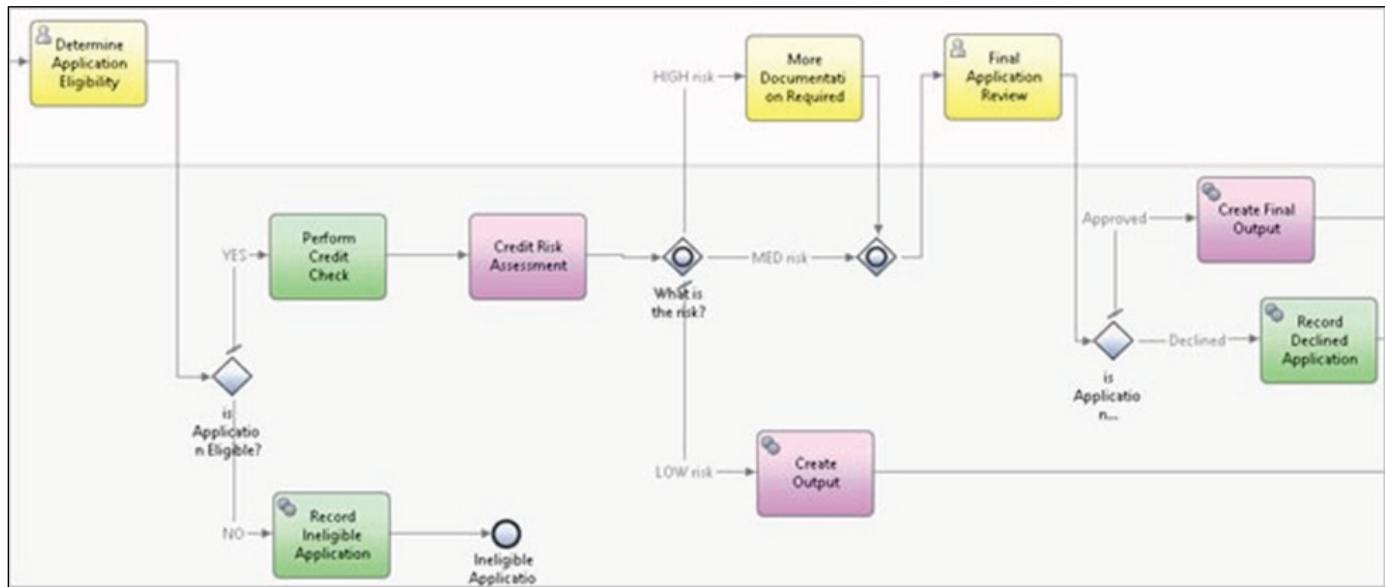
11. Right-click the **requestAccountAmount** field and click **Show In > Properties View** from the menu.
12. In the Properties view, click **Browse** beside the **Type** field.
13. Select **int** from the list and click **OK**.



14. Press **Ctrl + S** to save your changes and **Ctrl + W** to close the editor.

Part 2. Implement interfaces that use business objects as inputs and outputs.

In this portion of the exercise, you create shared interfaces that your modules use. Because the components you create in subsequent exercises use the same business object inputs and outputs, you can reuse the interfaces by defining them in a shared library. To save time, many of the shared interfaces were created for you in the workspace.



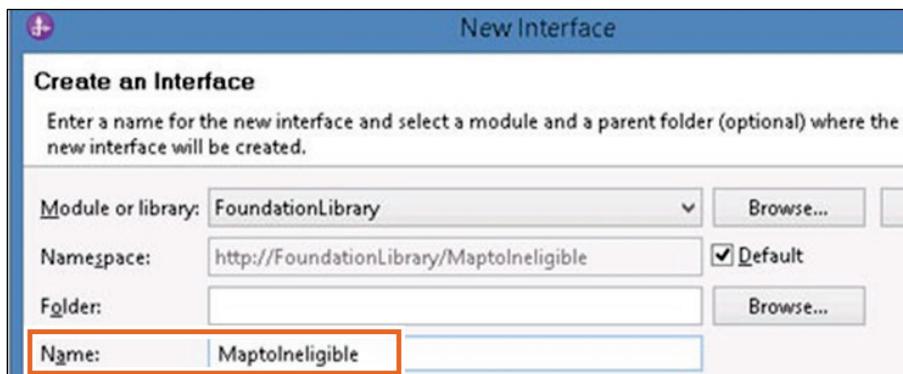
In this exercise, you create two interfaces:

- **MaptoIneligible:** This interface takes the customer's application as its input and produces an ineligible application as the output. The implementation of an activity does the transformation.
- **RecordIneligibleApplication:** This interface takes an ineligible application and returns a message. The implementation of an activity records the ineligible application.

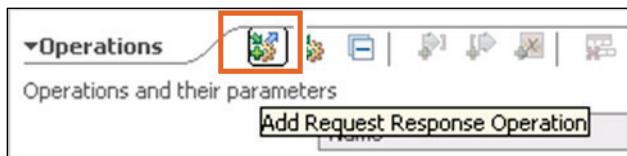
1. Create the shared interfaces.

Create an interface `MaptoIneligible` with a request and response operation that is named `InputCriterion` that uses the **CustomerApplication** business object as the input (named `Input`), and **IneligibleApplication** as the output (named `Output`).

1. Expand **FoundationLibrary**, right-click **Interfaces**, and click **New > Interface** from the menu.
2. In the “Create an Interface” dialog box, type `MaptoIneligible` in the **Name** field.



3. Accept the remaining default options and click **Finish** to open the **MaptoIneligible** interface in the interface editor.
4. In the interface editor, click the **Add Request Response Operation** icon.



5. Change the default operation name from `operation1` to: `InputCriterion`
6. Change the default input name from `input1` to `Input` (note the case).
7. Change the output name from `output1` to `Output` (note the case).

Name	Type
InputCriterion	
Inputs	Input
Outputs	Output

8. Change the default input type by clicking the **string** link and selecting the **CustomerApplication** business object from the menu.

Name	Type
InputCriterion	
Inputs	Input
Outputs	Output

A dropdown menu is open over the 'string' link in the 'Type' column. The menu lists various data types, with 'CustomerApplication' highlighted and enclosed in a red box.

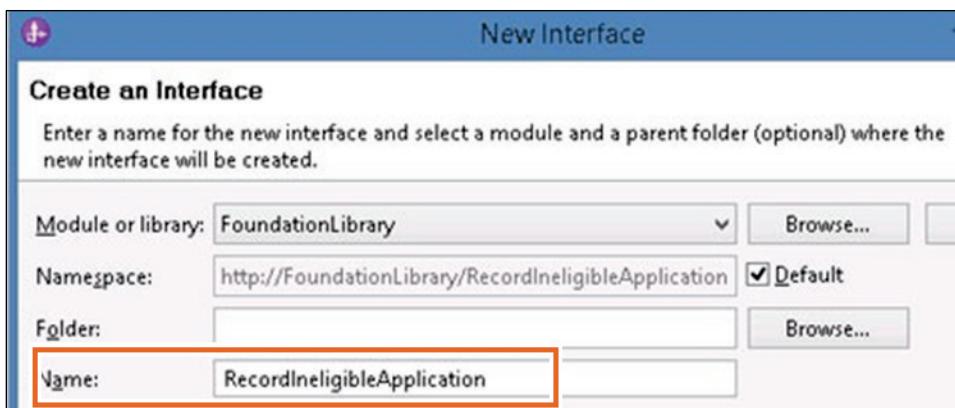
9. Click the link for the **Output** type and select the **IneligibleApplication** business object from the menu.

Name	Type
InputCriterion	
Inputs	Input
Outputs	Output

The 'Output' row is selected. A dropdown menu is open over the 'CustomerApplication' link in the 'Type' column. The menu lists various business objects, with 'IneligibleApplication' highlighted and enclosed in a red box.

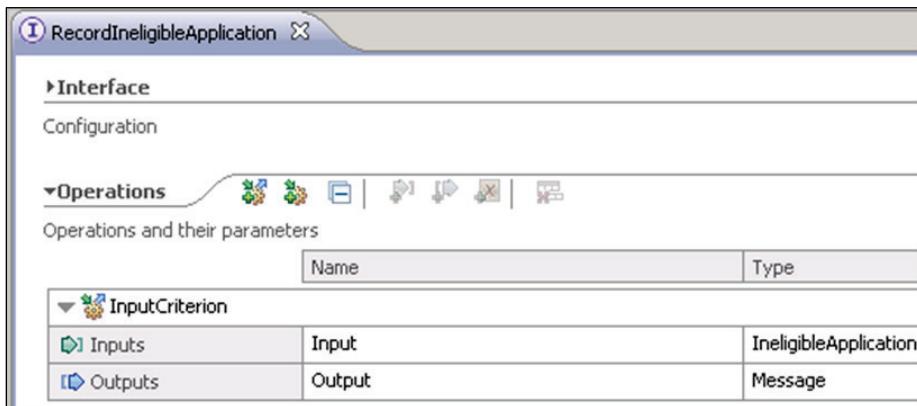
10. Press **Ctrl + S** to save your changes.
 11. Press **Ctrl + W** to close the editor.

2. Create another interface RecordIneligibleApplication with a request and response operation that is named InputCriterion that uses the IneligibleApplication business object as the input (named Input) and the Message business object as the output (named Output).
1. Expand **FoundationLibrary**, right-click **Interfaces**, and click **New > Interface** from the menu.
2. In the “Create an Interface” dialog box, type RecordIneligibleApplication in the **Name** field.



3. Accept the remaining default options and click **Finish**.
4. Click the **Add Request Response Operation** icon and change the operation name from `operation1` to: `InputCriterion`
5. Change the default input name from `input1` to `Input` (note the case).
6. Change the output name from `output1` to `Output` (note the case).
7. Change the default input type by clicking the **string** link and selecting the **IneligibleApplication** business object from the menu.
8. Click the link for the **Output** type and select the **Message** business object from the menu.

The completed interface resembles the following figure:



9. Press Ctrl + S to save your changes.

10. Press Ctrl + W to close the editor.

If time permits, examine the remaining business objects and interfaces that are created for you. In the remaining exercises, you build components that are going to use these artifacts.

11. Close IBM Integration Designer.

Results:

In this exercise, you created business objects and interfaces that the components use in your end-to-end solution.

References

- OSOA: Service Component Architecture family of specifications
 - <http://oasis-open.org/sca>
- developerWorks topics about Service Data Objects (SDO)
 - <http://www.ibm.com/developerworks/java/library/j-sdo/>
- Service Data Objects
 - <http://www.oasis-open.org/sdo>
- XML schema
 - <http://www.w3.org/XML/Schema>

Unit 8 Business process choreography overview

The slide features a blue header bar with 'IBM Training' on the left and the IBM logo on the right. The main content area has a light gray diagonal striped background. The title 'Business process choreography overview' is centered in large blue text. Below it, the text 'IBM Business Process Manager V8.6' is displayed in smaller blue text. At the bottom, a copyright notice reads: '© Copyright IBM Corporation 2018' and 'Course materials may not be reproduced in whole or in part without the written permission of IBM.'

IBM Training

**Business process
choreography overview**

IBM Business Process Manager V8.6

© Copyright IBM Corporation 2018
Course materials may not be reproduced in whole or in part without the written permission of IBM.

Unit objectives

- Describe the purpose and business value of using the WS-BPEL standard
- Describe the function of the business process container
- Describe the difference between long-running and microflow (short-running) business processes
- List and describe the seven parts of a business process

Unit objectives

This unit teaches you about SCA components and their bindings. You also learn how to use and implement web services in the SCA framework, specifically with IBM Integration Designer.

Topics

- Introduction to Web Services Business Process Execution Language (WS-BPEL)
- Elements of BPEL processes

Introduction to Web Services Business Process Execution Language (WS-BPEL)

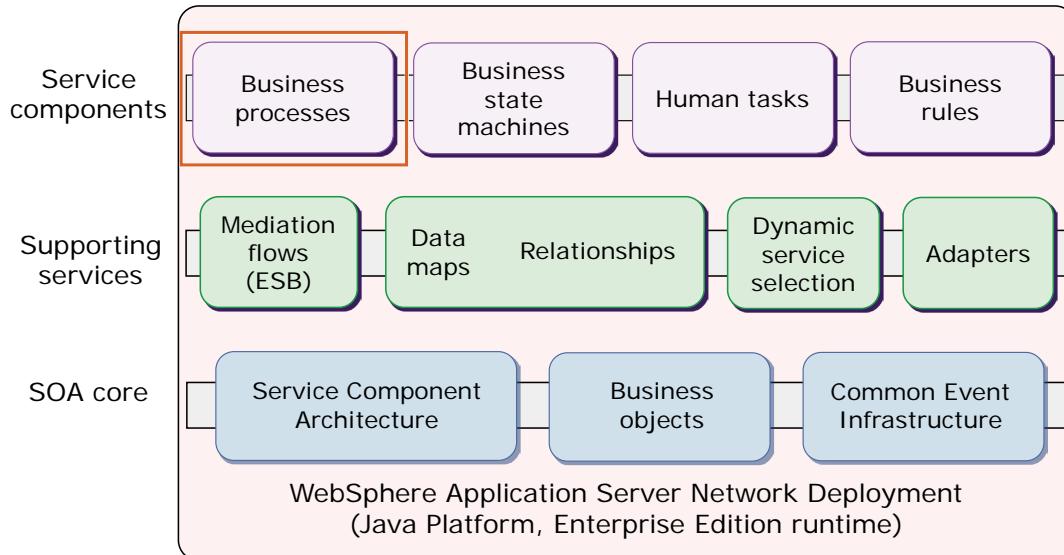
Business process choreography overview

© Copyright IBM Corporation 2018

Introduction to Web Services Business Process Execution Language (WS-BPEL)

Business processes are service components

- Business processes are part of the service components layer



Business process choreography overview

© Copyright IBM Corporation 2018

Business processes are service components

WS-BPEL (Web Services Business Process Execution Language) is the language for composing business processes. Processes that are created with the business state machine editor and with the business process editor are deployed as BPEL processes on IBM Process Server. Processes and state machines are part of the “service component” layer. BPEL (business process or state machine) is one of the implementation types for an SCA component.

Business processes in IBM Business Process Manager

- Business processes might be captured in IBM Process Designer or in IBM Integration Designer
- IBM Process Designer
 - Captured as business process
 - Represented as business process diagram (BPD)
 - Implementations that are captured in one process application or toolkit only
 - Limited implementation options
 - One human task client
- IBM Integration Designer
 - Captured as BPEL process
 - Represented with Business Process Execution Language (BPEL)
 - Implementation might span several modules
 - Might be shared in libraries
 - Several implementation options
 - Several human task clients

For differences between BPD and BPEL, and when to use which, see Unit 2

Business processes in IBM Business Process Manager

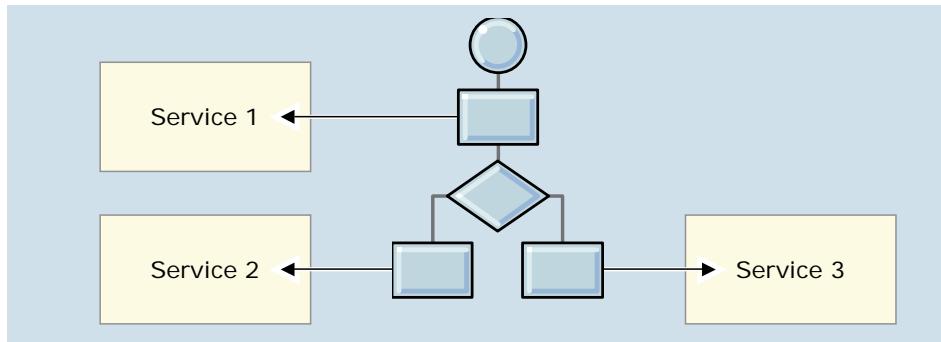
Business processes can be captured in either IBM Process Designer, IBM Integration Designer, or both. However, differences exist between the two tools:

- IBM Process Designer
 - Business processes are captured as business process artifacts based on Business Process Model and Notation (BPMN). They must be built in either a process application or a toolkit.
 - Business processes do not use a standard language, such as BPEL.
 - The only way to share a process is to place it in a toolkit.
 - Implementation options are limited (human tasks, JavaScript services, simple business rules, and others).
- IBM Integration Designer
 - Business processes are captured as BPEL artifacts. They can be built in either a module or a library.
 - They use the standard BPEL with IBM extensions (WS-BPEL).
 - They are loosely based on BPMN.

- Business processes can rely on services in other modules through imports and exports, or through libraries.
- BPEL processes in modules can be exposed as other types of services through their bindings (such as web services and WebSphere MQ bindings).
- A BPEL process can have several types of implementations, for example: including the full power of Java, integration with JRules, and integration with external services.
- Several types of human task clients are available, including HTML-Dojo in Business Space, JavaServer Faces, and the BPEL Process Choreographer Explorer.

Overview: Business Process Execution Language (1 of 2)

- A business process is a flow of execution paths that are described in WS-BPEL, including:
 - Which services are invoked
 - In what order services are invoked
 - The movement of data between services
- BPEL facilitates the building of composite integration applications by allowing the reuse of existing IT assets that are exposed as services



Business process choreography overview

© Copyright IBM Corporation 2018

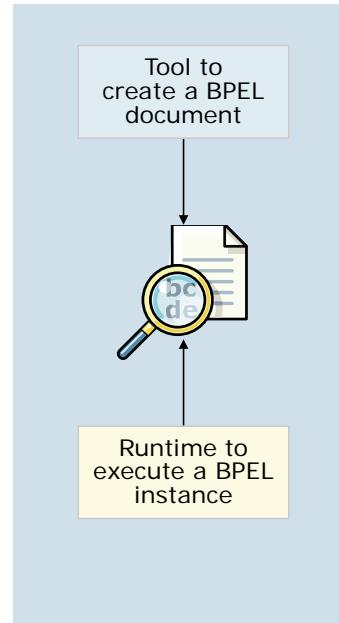
Overview: Business Process Execution Language

The business process component of IBM Process Server is arguably the most important aspect of the solution. This component provides the engine and management to run the steps for each instance of a process. Process orchestration occurs within the engine, and like the conductor of an orchestra, the process engine guides all the other components.

To further illustrate, this diagram shows a set of services that provide their “black box” functions, in which only their externally visible behavior is considered and not their implementation or inner workings. To build a solution, you design a process that choreographs their execution. This process is described in the BPEL language. The core concepts that must be captured are: which services are used, in what order they are called, and what data is passed from one service to another.

Overview: Business Process Execution Language (2 of 2)

- Implementation in BPEL decouples the process from the runtime engine, making the processes vendor-neutral and technology-neutral
 - Other companies provide products with BPEL runtimes
- BPEL supports the required technology patterns for business
 - For example: Error handling, compensation, and asynchronous processing
- The BPEL-compliant business process container included with IBM Process Server manages and runs business processes
 - Runs complex business processes securely, consistently, and with transactional integrity
 - Provides high performance and quality of service
 - Provides fault tolerance and error-detection capability



Business process choreography overview

© Copyright IBM Corporation 2018

When you create a business process, you create a BPEL document that describes the choreography of your process, and you use an appropriate runtime environment to run instances of the process. You need a tool to create the BPEL description. In IBM Integration Designer, a business process editor is provided. BPEL is detailed and is described in XML. As a result, the code is not easily readable to humans. It would be difficult to compose a BPEL process by using a simple XML editor.

When you create the BPEL document, it exists independently from the tool that created it. This independence is the portability aspect of BPEL. It is not coupled to any particular choice of implementation if the vendor complies with the specification.

An OASIS technical committee manages the WS-BPEL specification (currently at V1.1 with V2.0 now available): <http://www.oasis-open.org/home/index.php>

The specification is based on WSDL and other XML standards:

- Interfaces are described by using WSDL.
- XML schema and XPath expressions define data context handling.

BPEL4WS was version 1.1 of the BPEL specification. When version 2.0 of the specification was introduced, the open source committee renamed the specification to WS-BPEL. In general, people refer to the specification as BPEL. It is an XML-based language that is used to define the flow of a business process.

The WS-BPEL standard allows business process models to be defined independent of the implementation, keeping the processes separate from the underlying infrastructure or technology. This independence fits nicely with the concept of a service-oriented architecture (SOA) where interfaces are kept separate from implementations. WS-BPEL uses services and service interfaces as a means of defining the connections between the different steps.

For example, in a business process with five steps, interfaces on the steps indicate the type of data that is passed and potentially received and the type of operation to do. The WS-BPEL standard uses other industry standards such as Web Services Description Language (WSDL) to define steps and interfaces. XML Schema Definition (XSD) is used to define data structures. The BPEL process is an XML file that is interpreted at run time to indicate the sequence of steps that make up a business process. XPath support is also provided as a primary means of working with data objects that are passed between steps.

The BPEL specification and its concepts have existed for several years. Originally started by IBM, BEA, and Microsoft, the specification is now more refined; SAP and Siebel (and others) support it as well.

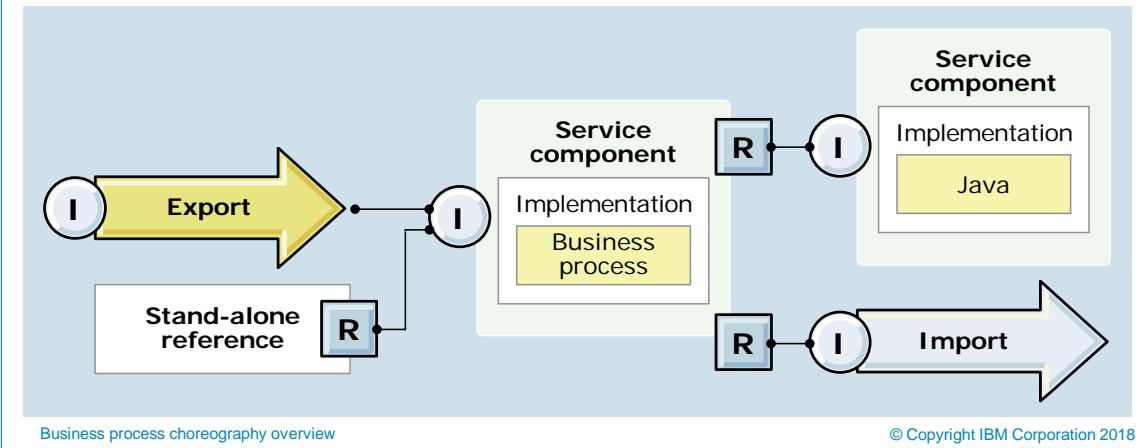
BPEL defines three separate parts or phases of a business process architecture:

- The development of a business process model
- The implementation of the activities that are involved in the business process
- The runtime engine that runs the business process model by determining the correct path through the business process and calling the necessary activities

External partners are invoked as web services. The BPEL business process is also available as a web service, so it can potentially become an activity or subprocess of a larger process.

BPEL interoperates with SCA and SDO

- BPEL processes can invoke other SCA components
- BPEL processes can be invoked as SCA components
- Partner Links are resolved to SCA components or external services
- SDO provides a standard data format for messages
 - SDO messages are sent and received



BPEL interoperates with SCA and SDO

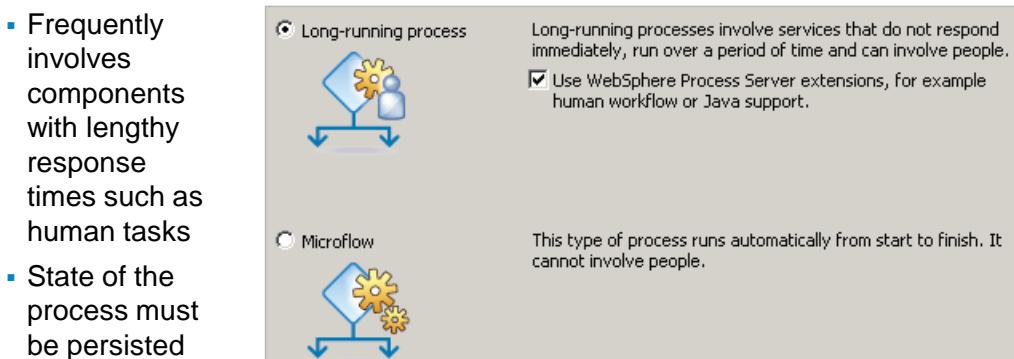
BPEL is an SCA implementation type. Therefore, business processes are exposed as service components, and the caller is unaware that the service is rendered in BPEL. The interface (interface partner) provides the information of how to invoke the BPEL (operations that are provided and the types of inputs, outputs, and fault messages). Processes can be invoked as SCA components.

BPEL implements reference partners and interface partners (called partner links) that are converted into SCA component interfaces and references.

SCA and SDO are used for internally doing the steps of the business process; they represent the data and how clients can work with business processes. Clients can easily invoke a business process as an SCA component and pass data to the process in the form of business objects (SDO). The way that business processes work with other services is also defined in SCA. You can create a service component with a BPEL implementation and wire it to a partner link (if it exists in the same module). Or you can wire the component to an import (if the partner exists in another module). Business processes can also be exposed to the outside world by creating an export component or by using a stand-alone reference (if a non-SCA component in the same module calls the business process).

Microflows versus long-running processes

- Business processes run on IBM Process Server as either microflows (short-running) or long-running processes
 - Known as non-interruptible and interruptible processes
- A microflow process is used for running short business processes or small units of work within a larger business process (subprocesses)
 - Microflows complete or fail and are not persisted
- A long-running process might run for hours, days, or weeks



Microflows versus long-running processes

The first type of business process is a non-interruptible process that is used for running short business processes or a small unit of work within a larger business process (a subprocess). A microflow business process either completes or fails. No intermediate state can be maintained.

Consider a credit card verification process. This business process is short and simple: accepting personal information and returning a result, which indicates whether the credit card was valid. In this business process, it might start with a number of lookups that check information about the individual before returning the results. These lookups either succeed and return a result, or fail and the entire credit card validation fails. The client can monitor the failure, and can call the business process again. In any case, no state information is being maintained in the different lookup steps.

The second type of business process is long-running (interruptible). These business processes might take hours, days, or weeks to complete, and because of the possible random completion time, state must be maintained during execution. If state is not maintained and the current execution thread that is running the business process ends (for example, if the server fails), the work that is accomplished would be lost.

An example of a long-running business process is a loan approval process. A loan approval process gathers personal financial information and evaluates it. If the individual is financially sound, a loan can be offered and the funds can be reserved.

Because the evaluation of the financial information can take a number of days, the state of this business process must be maintained. After each activity in the process is completed, the results must be recorded to follow which information was processed and how much work was completed.

When you create a business process in IBM Integration Designer (either long-running or microflow), you must elect whether to use IBM extensions to BPEL in the process. To disable the extensions, clear the **Use IBM Process Server extensions** check box in the new business process wizard. If you decide to turn off the extensions, you lose the following functions:

- **Process type**
 - Microflow
- **Expression language**
 - Java
- Actions
 - Human task
 - Snippet
 - Generalized flow
 - Collaboration scope
- **Properties for all activities (including the process)**
 - Description
 - Documentation
 - Display name
 - Custom properties (not applicable for structures)
 - Enable persistence and queries of business-relevant data
- **Process properties**
 - Auto-delete
 - Autonomy (whether a process runs as a peer or as a child of the invoking process)
 - Compensation sphere
 - Valid from
 - Ignore missing data (use this option to suppress runtime faults when data is missing during assignments)

- **Extensions for specific activities**
 - Scopes that can be flagged as non-compensable
 - Transactional behavior and the “continue on error” function of the invoke activity
 - Compensation of the invoke activity
 - Expiration setting on the invoke activity
 - Administrative tasks for processes, invokes, and snippets
 - Authorization tasks for receive, OnMessage, and OnEvent
 - Query properties on variables
 - Use of data type variables on the details tab of messaging activities like invoke, receive, reply, OnEvent, OnMessage

The following lists some cases when you would not want the extension enabled:

- When you are designing the process to use or edit in another set of tools
- When you are planning to run the process in a runtime environment other than IBM Process Server
- When you want to exchange information with a business partner who is not using the IBM Integration Designer set of tools or the IBM Process Server runtime environment

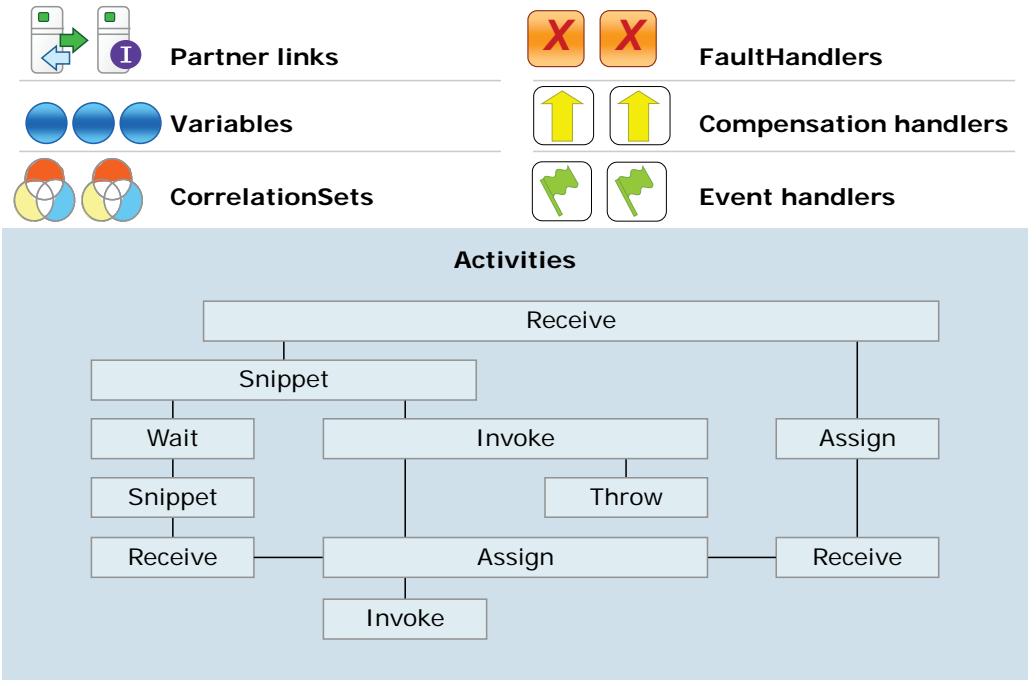
Elements of BPEL processes

Business process choreography overview

© Copyright IBM Corporation 2018

Elements of BPEL processes

Main elements and concepts of a BPEL process



Business process choreography overview

© Copyright IBM Corporation 2018

Main elements and concepts of a BPEL process

Elements of a business process defined

Icon	Element description
	1. Partners define parties that interact with the process
	2. Variables specify information that is used while running the business process
	3. Correlation sets match messages to the correct process instance
	4. * Fault handlers recover from partial and unsuccessful work that is done in the current scope of the business process
	5. * Compensation handlers contain actions that do reverse operations for a particular scope or activity
	6. * Event handlers do work that is based on an event or an asynchronous message
Varied	7. * Activities are used to define the process logic

Business process choreography overview

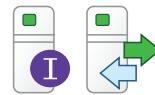
© Copyright IBM Corporation 2018

Elements of a business process defined

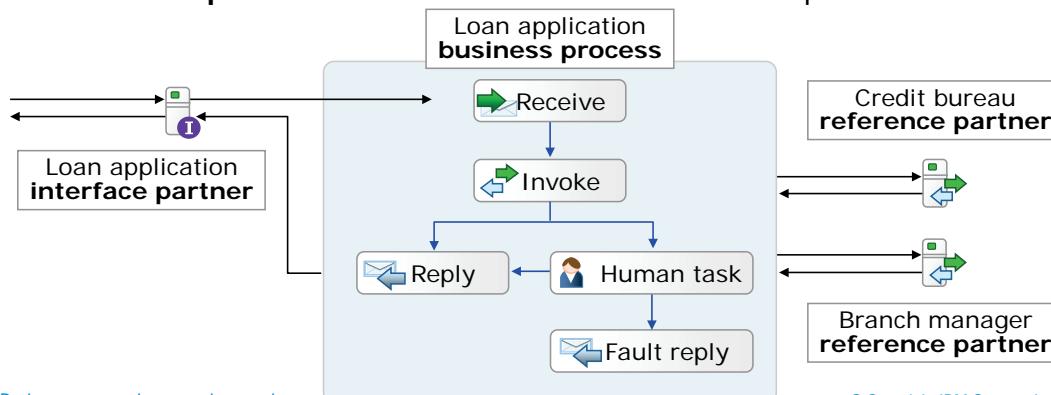
The BPEL specification defines the following seven primary elements or concepts:

- Partner links (partners)
- Variables
- Correlation sets
- Compensation handlers
- Event handlers
- Activities
- Fault handlers

Process elements: Partners



- **Partners** represent the external entities with which your business process interacts
 - An interface file describes the operations that the partner advertises
- The business process engine recognizes two categories of partners:
 - **Interface partners** define the services that the business process provides, which clients can invoke
 - **Reference partners** define the services that the business process can invoke



© Copyright IBM Corporation 2018

Process elements: Partners

A partner is an abstract entity that represents an external service with which the business process interacts. The WS-BPEL specification introduces the notion of a partner link to describe a reference to one of these entities. IBM Integration Designer automatically creates the partner links for you when you create a partner.

The WS-BPEL specification does not differentiate between interface and reference partners. Both types of partners exist in the same category. A partner link might have both an interface and a reference role. Certain operations represent entry points into the business process, while other operations represent outbound calls from the process to other services.

In SOA, each service becomes a building block of a larger service. To define the service interaction point, a partner link component is necessary. This page introduces the concept of an interface partner (where data is coming into this business process) and a reference partner (where data gets sent to an external service for processing).

Process elements: Variables



- **Variables** hold data that represents the state of a process
- Variables:
 - Can be internal, received from partners, or sent to partners
 - Can be inputs or outputs for Invoke, Receive, and Reply activities
 - Can be manipulated from Assign and Snippet activities
 - Can be global (process-level) or local
- Variables are typically associated with WSDL message types
 - XSD data types are also available
- Variables are tied to scopes (containers of one or more activities)
 - Initialized by the runtime per the scope before they can be used
 - You can control the initialization order for multiple variables
- Global variables:
 - Are available to the process and embedded scopes
 - Can be queried at run time
- The process tray shows both global variables and variables that are defined in the selected scope (local variables)
 - Can drag variables from the tray onto activities to assign inputs and outputs

Process elements: Variables

Variables represent the state of the business process. Variables store information about data that is coming into the business process, which is then reused and sent to the various steps that are run as part of the business process. Variables are also used to hold the data that is returned from a service back into the business process or to hold internal information such as counter values for iterative processes. The variable capability enhances in line with the BPEL specification to allow the use of XSD data types. WSDL message types can also be used to facilitate the service-oriented architecture that BPEL builds upon. The business process steps are defined as an interface that has messages, and those messages can be used as the basis for your variable types.

After the business logic, one of the most important aspects of building a business process is working with variables and establishing the state. It includes how the state is used and how information is passed from one activity to another. In IBM Process Server, variables must be initialized in the business process engine or explicitly initialized by the developer.

Variables can be initialized in the business process engine by specifying the variable as the destination for a receive activity, a receive choice activity, or an event. With the receive activity, information comes in the form of a message, which is stored in a variable. The business process engine initializes that variable, and you can use it later. You can also take parts of the variable and move them to another variable. To use the second variable, you must first move some data into it or explicitly instantiate it.

In WS-BPEL V1.1, the XPath extension function `GetVariableData()` is used to access process variables. In WS-BPEL V2.0, XPath variable references (the `$variable` notation) are used to access elements of process variables or the status of a link. XPath variable notation applies to all XPath expressions and queries in assign activities or for all kinds of conditions. This notation represents a syntax change only. The WS-BPEL 1.1 notation is still supported.

You can set initialization values for BPEL variables and initialize complex business objects with the XML Literal option. To easily initialize variables that point to complex business objects, use the Value Composer, which is part of the assignments in an assign activity. You can initialize variables according to BPEL V2.0 specification. You can move variables up or down in the BPEL editor tray to control the order of initialization since ordering of variables when they are initialized can make a difference in how the application runs. You can drag variables onto invoke, receive, reply, and human task activities to assign inputs and outputs. In addition, you can drag variables onto collaboration scope activities to set the folder variable if the variable is of type caseFolder. You can drag a variable onto a forEach activity if the variable is of type array, and you can drop a variable onto a throw activity if the variable is a fault variable.

One of the primary ways to work with variables is the assign activity. An assign is used to move data from one variable to another and to manipulate data. For more complex operations, visual or text snippets can be used. Visual snippets offer many visual constructs that you can use to work with variables and operations from a predefined set of library functions.

Visual snippets generate Java code for you, simplifying the process of working with variables by reducing the need for integration specialists to understand low-level details. Text snippets, or “text mode,” uses Java programming and data object APIs to work with variables.

Using the variables tray, you can drag variables onto invoke, receive, reply, and human task activities to assign inputs and outputs. You can drag variables onto collaboration scope activities to set the folder variable if the variable is of type caseFolder. You can drag variables onto forEach activities if the variable is of type array, and you can drag variables onto throw activities if the variable is a fault variable.

Process elements: Correlation sets (1 of 2)



- **Correlation sets** route an incoming message to a long-running business process instance
 - Messages must contain a set of variables and values that uniquely identify the contents
- A correlation set has a name and is composed of correlation properties
 - Each property consists of a name and a data type
 - Message contents are mapped to correlation properties
- At run time, values in the message determine the business process instance to which that message is routed
 - The unique values need to be initialized the first time the correlation set is used
- Correlation sets can be specified for processes or for individual activities

Process elements: Correlation sets

Correlations direct requests to a business process instance, and long-running processes require them. For example, if 1000 loan applications are “in flight” and the credit department is sending credit report messages for multiple credit applications, correlation sets ensure that the messages are sent to the correct instances. Within a correlation set, you define the parts of an incoming message that indicate uniqueness. These parts then map to some type of value within the business process state. The messages can then be matched with an instance that contains the same unique message parts.

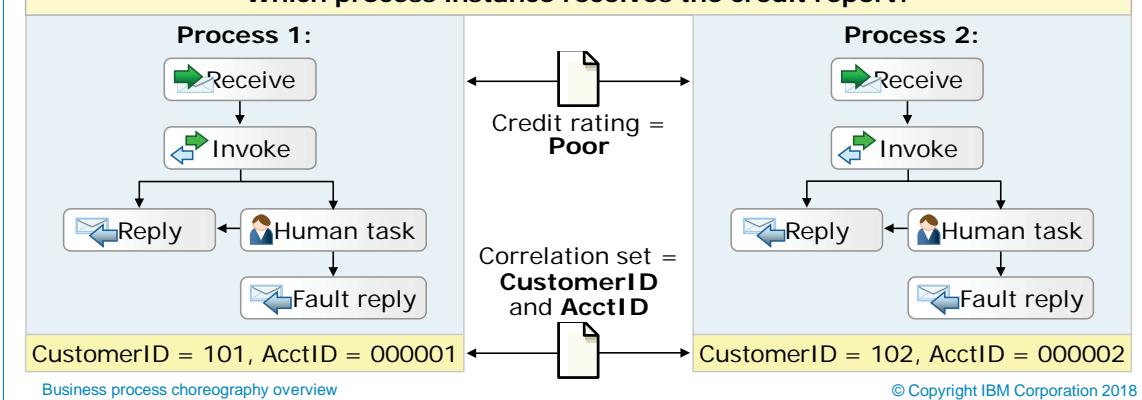
It is important to note that correlation sets are provided from unique values in application data. An example would be a social security number, order number, customer number, or some value that is defined as part of the business process. Correlation sets can consist of a single value or multiple values, and multiple correlation sets can exist as the long-running business process runs over a time period. As the business process state changes, different unique identifiers might be used to identify an instance. The correlation set must be initialized, typically at the beginning of a long-running business process.

Process elements: Correlation sets (2 of 2)

- Business processes maintain state information when interacting with partners
- The business process engine has the following characteristics:
 - Accepts an incoming message
 - Correlates the message elements to the parts in the correlation set
 - Compares the message values to the values in the set
 - The message is directed to the appropriate business process instance that matches the values in the set

A loan application process correlates client information when receiving credit reports.

Which process instance receives the credit report?



Correlation sets are used in runtime environments when multiple instances of the same process run. The sets allow two partners to initialize a BPEL process transaction, temporarily suspend activity, and then recognize each other again when that transaction resumes.

In the slide, two running instances of the loan application process exist. While both of the customers have a credit rating of “poor”, they need to recognize their own instance of the “poor” credit report. For the two customers to recognize their “own” process instances, the correlation set composed of CustomerID and AcctID uniquely associates the correct process to the correct customer. This association ensures that the correct credit rating reaches the correct customer.

The process choreography runtime environment does not generate the correlation ID (although the business process container does map messages to unique instances). The correlation ID is required at the time that the business process instance is started. Typically, the correlation ID value is some unique value that is known to the starter of the business process or is generated and returned to the starter before the business process is started. When the business process is started, the correlation ID is passed on the input message.

Consider a stock trading process where many different customers might be looking to buy or sell stocks. Each customer would contact the customer's broker with the intention to trade a particular stock. The broker would start an instance of the stock trade process with the customer number as the unique identifier. The customer number can be specified on the request and matched to a particular running instance.

Process administrators can repair correlation sets. Modeling or runtime errors can cause the initialization of a correlation set to fail even though the workflow engine needs the values. Business Process Choreographer Explorer uses new correlation set APIs and provides a repair page to correct this problem.

Process elements: Activities

- **Activities** are the individual business tasks that implement the larger business goal that the process represents
- An activity can be basic, structured, or associated with error processing
 - Basic activities do not contain other activities: human task, snippet, or reply
 - Structured activities are activities that contain other activities: sequence, or while loop
 - Activities for fault handling and compensation are activities that process expected and unexpected error conditions
- BPEL also uses handlers with certain activities
 - Handlers contain activities that are run as a result of events or during error processing

Unit summary

- Describe the purpose and business value of using the WS-BPEL standard
- Describe the function of the business process container
- Describe the difference between long-running and microflow (short-running) business processes
- List and describe the seven parts of a business process

Checkpoint questions

1. List the seven main parts of a business process.
2. Define the concept of a partner link.
3. What is the purpose of a correlation set?
4. What are the two types of business processes?

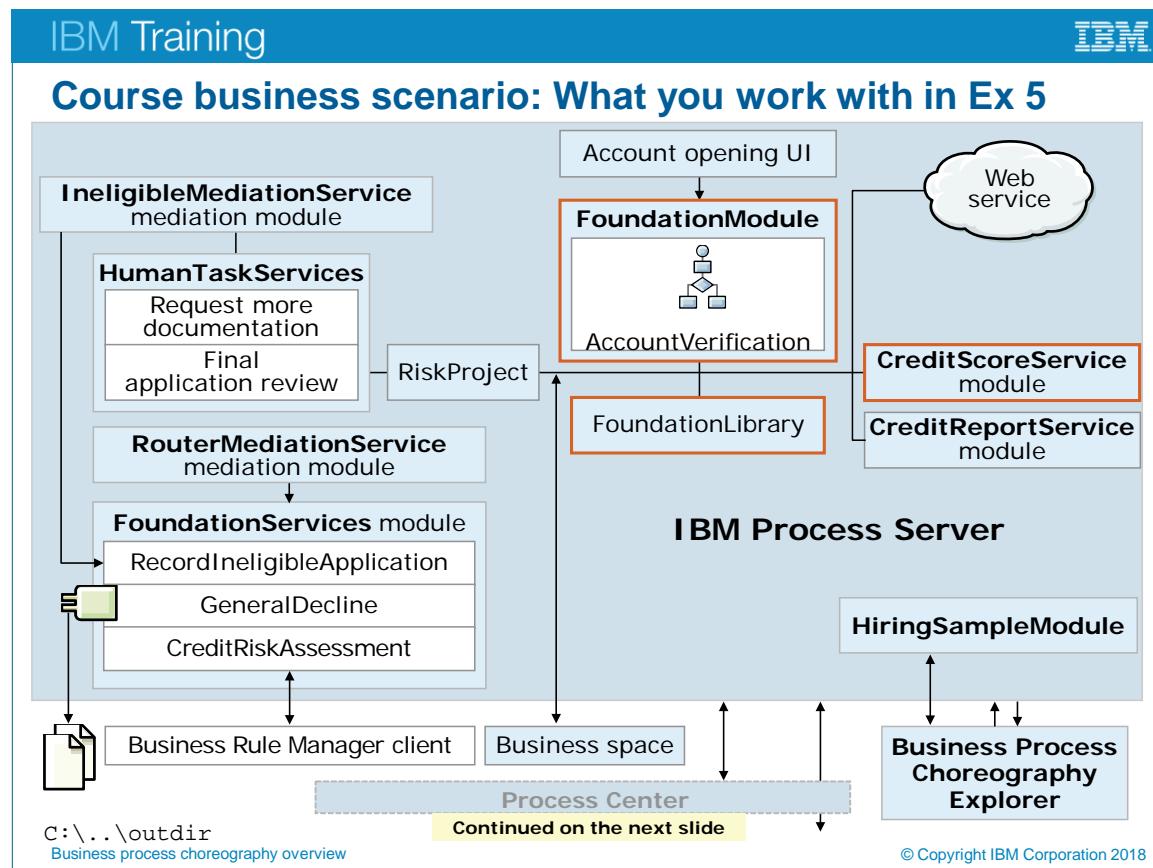
Checkpoint answers

1. Activities, fault handlers, event handlers, compensation handlers, variables, partners (partner links), and correlation sets.
2. Partners represent the external entities with which your business process interacts.
3. A correlation set routes an incoming message to the running business process instance that should handle it.
4. Microflows and long-running processes.

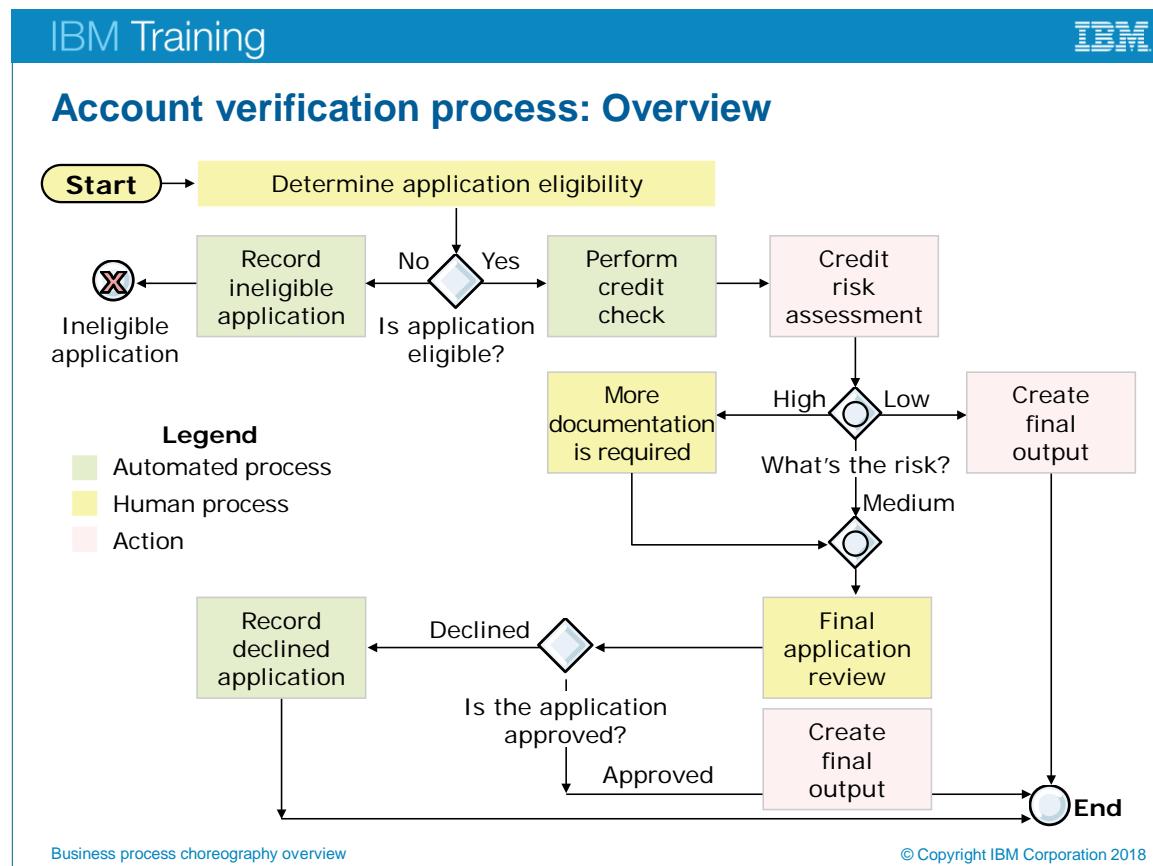
Exercise 5: Creating a business process, part I

After completing this exercise, you should be able to:

- Create a business process
- Implement WS-BPEL interface partners and reference partners
- Create process variables
- Compare business processes between IBM Integration Designer and IBM Process Designer



Course business scenario: What you work with in Ex 5



Account verification process: Overview

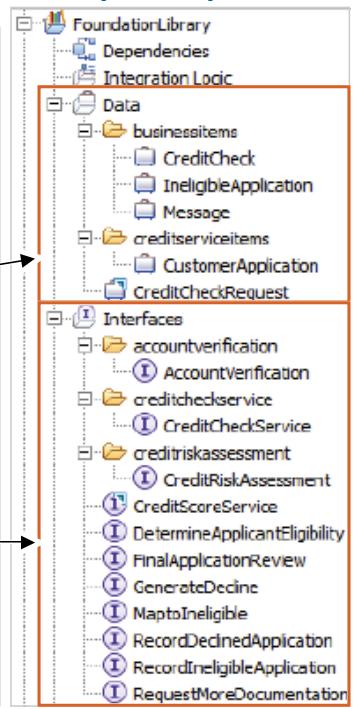
Why create the Account verification business process in IBM Integration Designer?

The business process application that you examined in IBM Process Designer in Exercise 1 is incomplete. Although the activities were in place, and in a later exercise you wired them together, each individual activity did not accomplish anything: they had no implementations. In this exercise and the subsequent exercises, you build this process in IBM Integration Designer; but more importantly, you focus on building the implementations for the individual activities.

Components that are required for Exercise 5 (1 of 7)

Prebuilt components that are imported in the lab:

- 1. FoundationModule**
- 2. CreditScoreService**
- 3. FoundationLibrary container for business objects:**
 - CreditCheck
 - IneligibleApplication
 - Message
 - CustomerApplication
 - CreditCheckRequest
- 4. FoundationLibrary container for interfaces:**
 - AccountVerification
 - CreditCheckService
 - CreditRiskAssessment
 - CreditScoreService
 - DetermineApplicantEligibility
 - FinalApplicationReview
 - GenerateDecline
 - MaptoIneligible
 - RecordDeclineApplication
 - RequestMoreDocumentation



Business process choreography overview

© Copyright IBM Corporation 2018

Components that are required for Exercise 5

An interface provides the input and output of a component. It is created independent of the internal implementation of the component. The AccountVerification interface dictates how to use the AccountVerification process. It specifies the operations that can be called and the data that is passed, such as input arguments, returned values, and exceptions on the AccountVerification process.

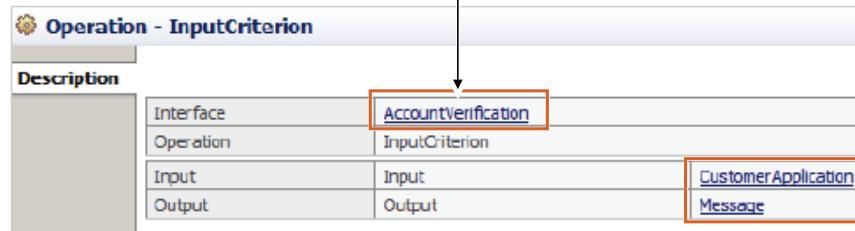
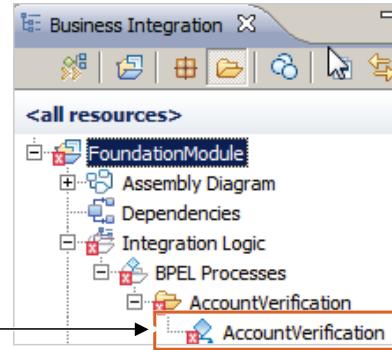
The interface partner is a direct link to the interface where the partner is configured. An interface partner is the process interface, and it exposes operations that external users or services can call.

Components that are required for Exercise 5 (2 of 7)

New components that you create in the lab:

1. AccountVerification BPEL process:

- Uses the **AccountVerification** interface



Components that are required for Exercise 5 (3 of 7)

New components that you create in the lab:

2. Global variables:

- CreditCheckVariable
- IneligibleApplicationVariable
- CustomerApplicationVariable
- CustomerApplicationVariable2
- MessageVariable



Variables
CustomerApplicationVariable
MessageVariable
CreditCheckVariable
IneligibleApplicationVariable
CustomerApplicationVariable2

The strength of IBM Integration Designer is to use the Service Component Architecture, so other implementations can be placed in separate modules. Because implementations are spread throughout other modules, it maximizes reusability, and it necessitates intercommunication and passing variables between modules.

In this exercise, you add the following global variables to the AccountVerification business process. These variables store business objects for manipulation by the process activities:

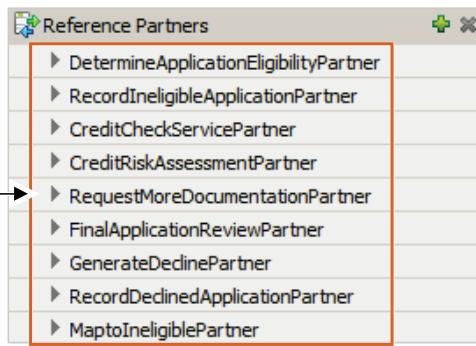
- The global variable that is named CreditCheckVariable stores a CreditCheckRequest business object.
- The global variable that is named IneligibleApplicationVariable stores an IneligibleApplication business object.
- The global variable that is named CustomerApplicationVariable2 stores a CustomerApplication business object.
- CustomerApplicationVariable and MessageVariable correspond to the type of business objects that form the input and output variables for the AccountVerification process interface.

Components that are required for Exercise 5 (4 of 7)

New components that you create in the lab:

3. Reference partners:

- DetermineApplicationEligibilityPartner
- RecordIneligibleApplicationPartner
- CreditCheckServicePartner
- CreditRiskAssessmentPartner
- RequestMoreDocumentationPartner
- FinalApplicationReviewPartner
- GenerateDeclinePartner
- RecordDeclinedApplicationPartner
- MaptoIneligiblePartner



In this portion of the exercise, you add WS-BPEL reference partners to your process. Reference partners represent the service interfaces that your WS-BPEL activities call during process execution.

1. Add the DetermineApplicationEligibilityPartner reference partner with the DetermineApplicantEligibility interface.
2. Add the RecordIneligibleApplicationPartner reference partner with the RecordIneligibleApplication interface.
3. Add the CreditCheckServicePartner reference partner with the CreditScoreService interface.
4. Add the CreditRiskAssessmentPartner reference partner with the CreditRiskAssessment interface.
5. Add the RequestMoreDocumentationPartner reference partner with the RequestMoreDocumentation interface.
6. Add the FinalApplicationReviewPartner reference partner with the FinalApplicationReview interface.

7. Add the GenerateDeclinePartner reference partner with the GenerateDecline interface.
8. Add the RecordDeclinedApplicationPartner reference partner with the RecordDeclinedApplication interface.
9. Add the MapToIneligiblePartner reference partner with the MapToIneligible interface.

IBM Training IBM

Components that are required for Exercise 5 (5 of 7)

New components that you create in the lab:

4. Interface partner

- **AccountVerification** interface partner

AccountVerification (AccountVerification)

- Interface Partners + ×
- ▶ AccountVerification
- Reference Partners + ×
- Variables + × (highlighted with a red box)
- Correlation Sets + ×
- Correlation Properties + ×

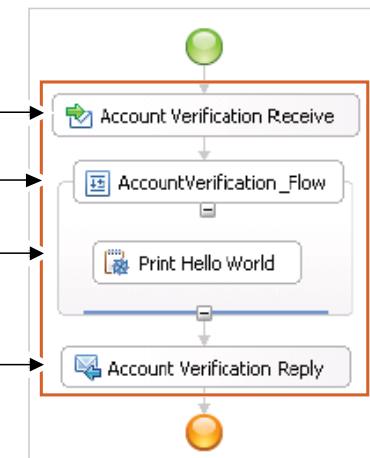
An interface provides the input and output of a component. It is created independent of the internal implementation of the component. The AccountVerification interface dictates how to use the AccountVerification process. It specifies the operations that can be called and the data that is passed, such as input arguments, returned values, and exceptions on the AccountVerification process.

The interface partner is a direct link to the interface where the partner is configured. An interface partner is the process interface, and it exposes operations that external users or services can call.

Components that are required for Exercise 5 (6 of 7)

New components that you create in the lab:

5. Receive activity
 - **Account Verification Receive**
6. Generalized flow activity
 - **AccountVerification_Flow**
7. Snippet action
 - **Print Hello World**
8. Reply activity
 - **Account Verification Reply**

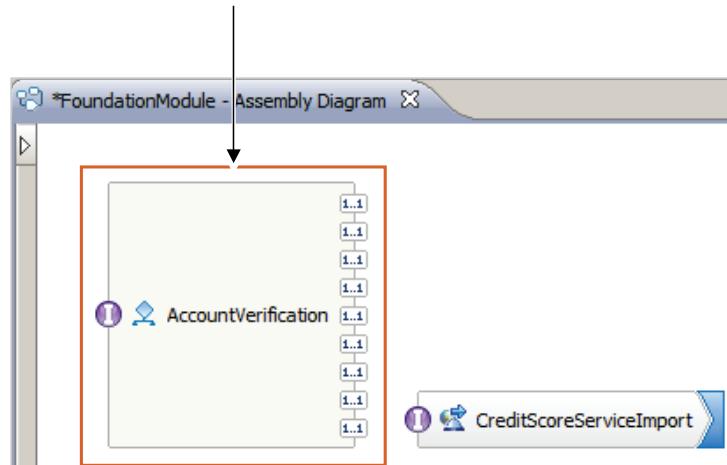


In this exercise, you add a generalized flow that is a structured activity that acts as the container for the simple and complex BPEL activities.

Components that are required for Exercise 5 (7 of 7)

New components that you create in the lab:

9. SCA component
 - **AccountVerification** component



Exercise 5: Creating a business process, part 1

Purpose:

In this exercise, you use IBM Integration Designer to begin creating a complex business process. You create a long-running business process, process variables, interface partners, and reference partners.

The Web Services Business Process Execution Language (WS-BPEL) provides syntax for specifying the behavior of a business process in a platform-independent manner. It is used to coordinate a series of service invocations to fulfill a business task. Although the language provides conditional and control logic, most of the work is done through the invoked services.

Combining WS-BPEL with the SCA programming model allows for the coordination of SCA services into much larger units of work. Individual SCA services can be brought together and can benefit from the advanced capabilities of event handling, fault handling, and compensation.

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.

Instructions

AccountVerification automates the process of opening customer accounts that are based on SOA principles. As a result, the process of opening customer accounts is more flexible (easily accommodating future changes), and existing services can be reused instead of rewritten.

In this exercise and the subsequent exercises, you build this process in IBM Integration Designer; but more importantly, you focus on building the implementations for the individual activities.

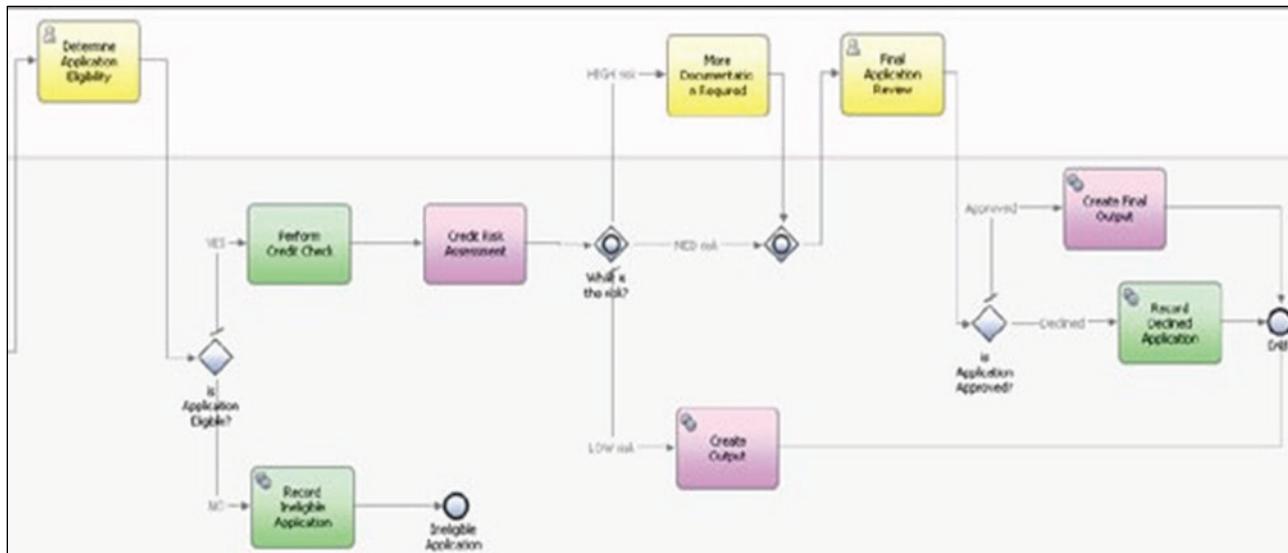
Part 1. Create a business process.

In this portion of the exercise, you create a long-running process that is named AccountVerification in FoundationModule. This process forms the core of the end-to-end application that you are developing in the exercises. The process is designed to receive a customer application, do a credit check, and send the application forward for approval, supplemental documentation, or rejection.

Review the process narrative that is used to create that process model: it serves as the narrative for your new business process.

- When the customer submits an application, the application must be tested for eligibility.
- If the application is ineligible, record the ineligible application in the database and end the process.
- If the application is eligible, the system calls an external service to do a credit check.
- A credit risk assessment is done against the customer's credit check.
- If the customer is determined to be low risk, the application is automatically approved. An output message is generated for the client, and the process is complete.
- If the customer is determined to be medium risk, the customer must seek final approval from an authorized figure for the application.
- If the customer is determined to be high risk, the customer must submit more documentation, and then the customer must seek final approval for the application.
- If the application is approved, generate an output message for the customer. The process is complete.
- If the application is denied, record the declined application. The process is complete.

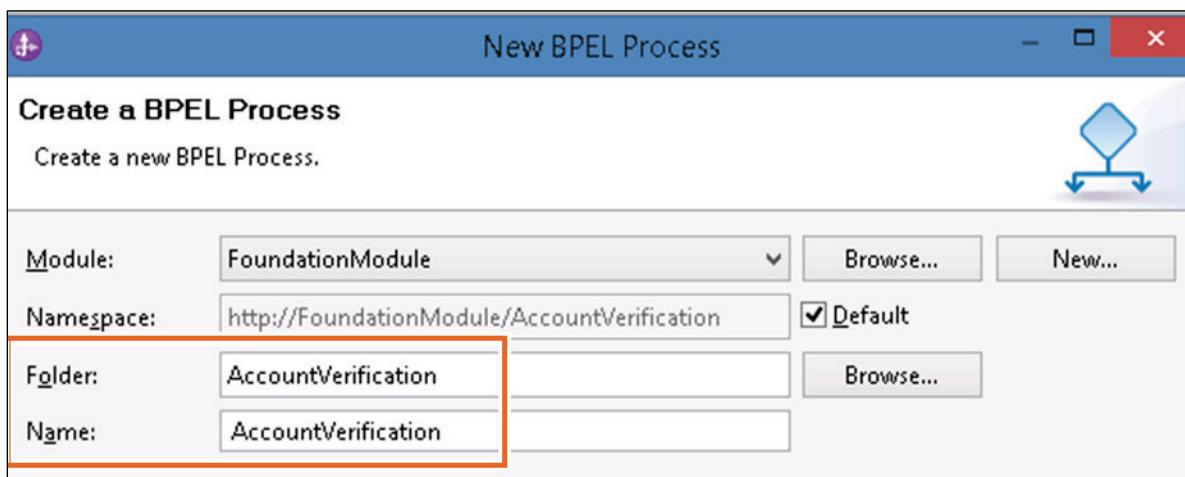
Do not be concerned about reading the small text in this diagram. The purpose of the solution diagram is to view the connection wiring and the flow.



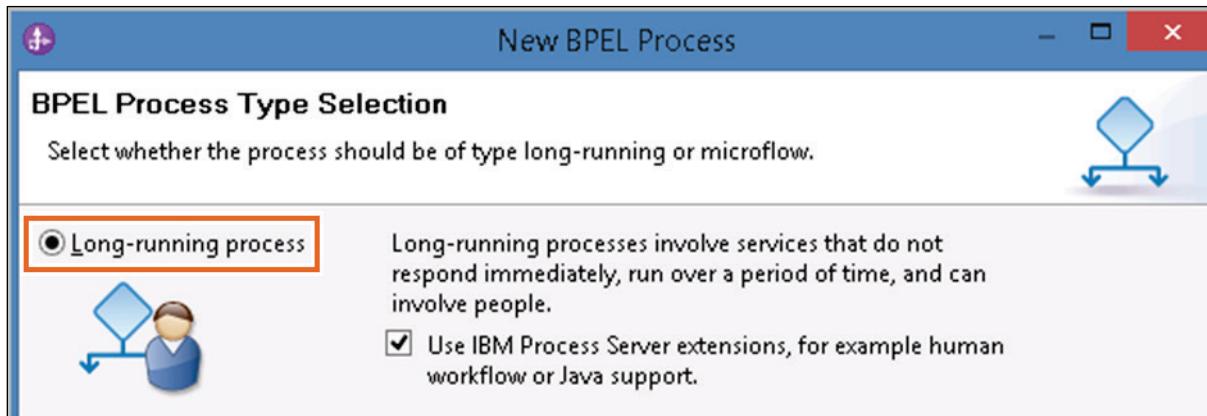
1. Create a long-running business process,

In this portion of the exercise, you create a business process in **FoundationModule** named **AccountVerification** that uses the **AccountVerification** interface

1. On your desktop, open the folder that is labeled **Exercise Shortcuts**.
2. Double-click the shortcut that is labeled **Exercise 5**. Allow Integration Designer a few moments to build the workspace. You can view the workspace build status at the lower-right corner of the 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 a previous exercise, you get this message.
4. Close the **Getting Started** tab.
5. Expand **FoundationModule**, right-click **Integration Logic**, and click **New > BPEL Process** from the menu.
6. In the New BPEL Process dialog box, at the “Create a BPEL Process” dialog box, enter the following information:
 - Type **AccountVerification** in the **Folder** field.
 - Type **AccountVerification** in the **Name** field.



7. Click **Next**.
8. At the BPEL Process Type Selection dialog box, verify that **Long-running process** is selected, and verify that **Use IBM Process Server extensions** is selected.

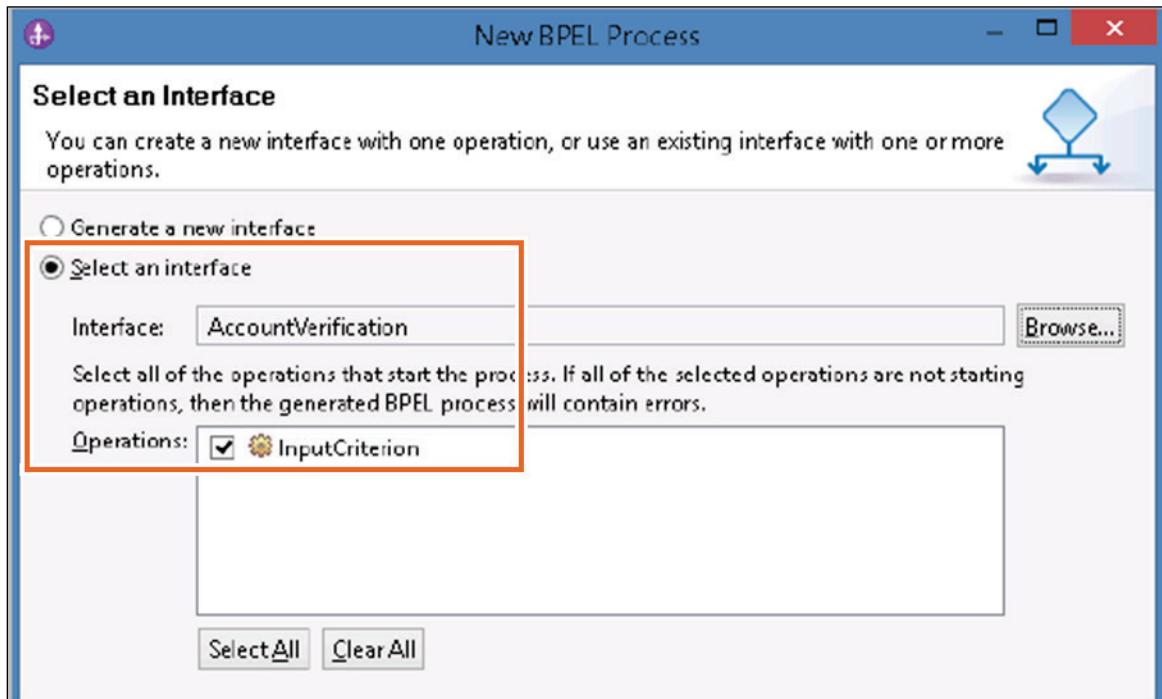


Note: For information about the capabilities that are lost by clearing the **Use IBM Process Server extensions** option, see the IBM Business Process Manager product documentation.

9. Click **Next**.

10. At the “Select an Interface” dialog box, take the following actions:

- Click **Select an interface**.
- Click **Browse**, select the **AccountVerification** interface from the list, and click **OK**.

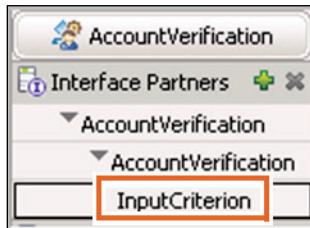


11. Accept the remaining default options and click **Finish** to open the process editor.

2. Examine the AccountVerification interface partner.

This interface becomes a WS-BPEL interface partner.

1. In the Interface Partners portion of the tray, expand AccountVerification > AccountVerification and select the InputCriterion operation.



2. Switch to the **Properties** view.

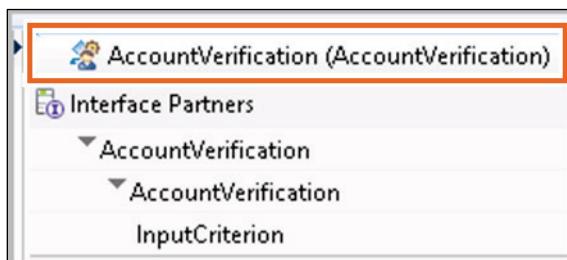
Note the properties of the interface and the business object inputs and outputs. The interface accepts a CustomerApplication business object as the input and returns a Message business object as the output.

Description	Interface	AccountVerification
	Operation	InputCriterion
	Input	Input CustomerApplication
	Output	Output Message

3. Examine the properties of the long-running process.

You learn more about these properties later in this course.

1. In the tray on the far right of the screen, click **AccountVerification (AccountVerification)**.



2. Switch to the **Details** tab in the **Properties** view.

3. Examine the detailed properties in the **General Settings** section.

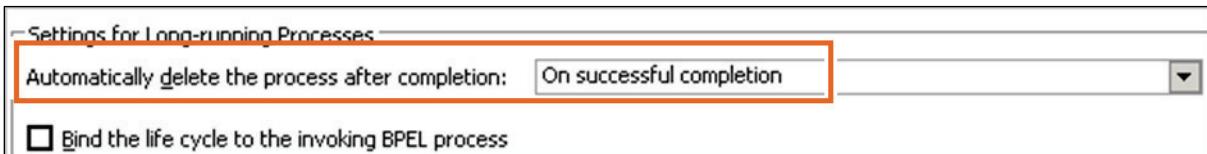
4. Note the value in the **Select the date when the process becomes valid** field.

Process instances that are created from this date forward use this process template. Processes currently “in flight” use the process template available when they were started unless you upgrade them to a new template.

Process - AccountVerification (AccountVerification)	
Description	General Settings
Details	BPEL process type: Long-running process
Server	Uses IBM Process Server extensions
Administration	<input checked="" type="checkbox"/> Select the date when the process becomes valid:
Java Imports	April 20, 2016
Defaults	Select Date

5. Examine the properties in the **Settings for Long-running Processes** section.

6. Note the value in the **Automatically delete the process after completion** field. By default, when a process is successfully completed, process-related data for the completed instance is deleted from the Business Process Choreographer database.



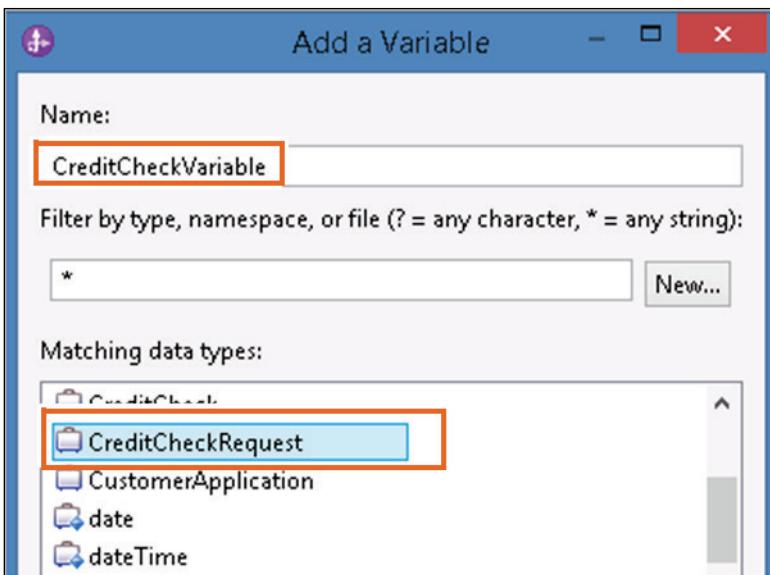
Part 2. Create process variables.

In this portion of the exercise, you add several global variables to the business process. The variables store business objects for manipulation by the process activities. A variable belongs to the scope in which it is declared. If it is declared in the global process scope, then it is a global variable and is visible to the process as a whole. If it is declared in a nested scope, it is called a scoped or local variable and is visible only by objects within the scope in which it is created.

1. Declare process variables.

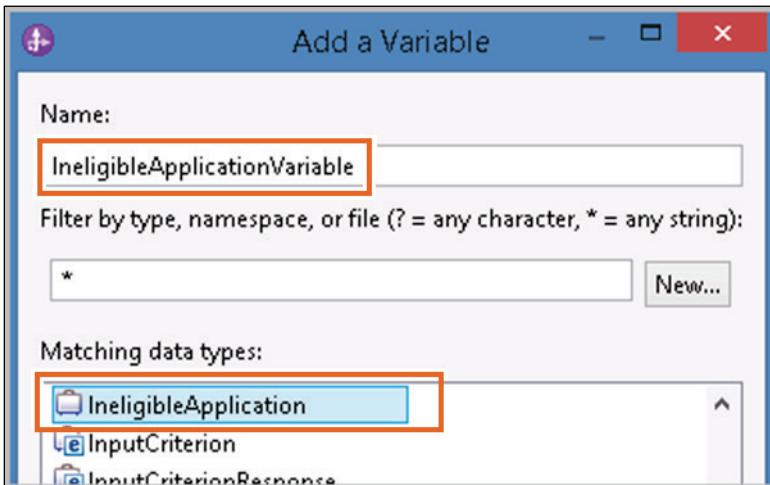
You add a global variable that is named `CreditCheckVariable` that stores a **CreditCheckRequest** business object.

1. In the **Variables** section of the tray, click the plus sign (+) to add a variable.
2. In the “Add a Variable” dialog box, change the **Name** to: `CreditCheckVariable`
3. In the “Matching data types” section, select the **CreditCheckRequest** business object.



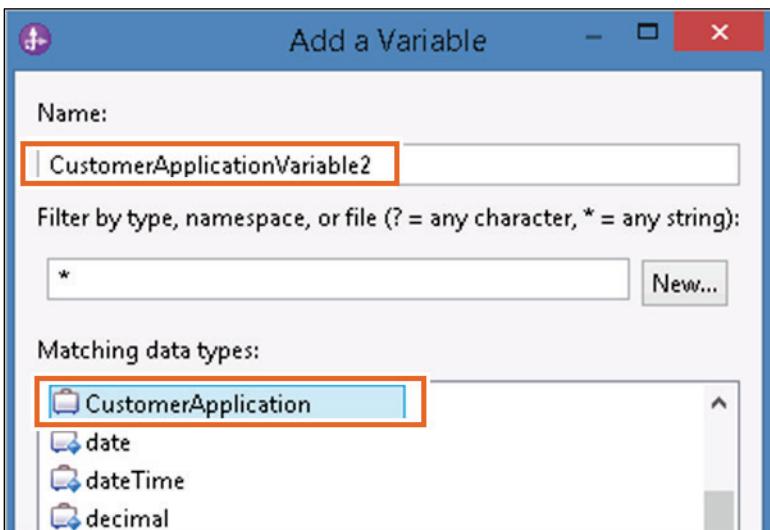
4. Click **OK**.

2. Add a global variable that is named `IneligibleApplicationVariable` that stores an `IneligibleApplication` business object.
1. In the **Variables** section of the tray, click the plus sign (+) to add a variable.
 2. In the “Add a variable” dialog box, change the **Name** to: `IneligibleApplicationVariable`
 3. In the “Matching data types” window, select the **IneligibleApplication** business object.



4. Click **OK**.
3. Add a global variable that is named `CustomerApplicationVariable2` that stores a `CustomerApplication` business object.

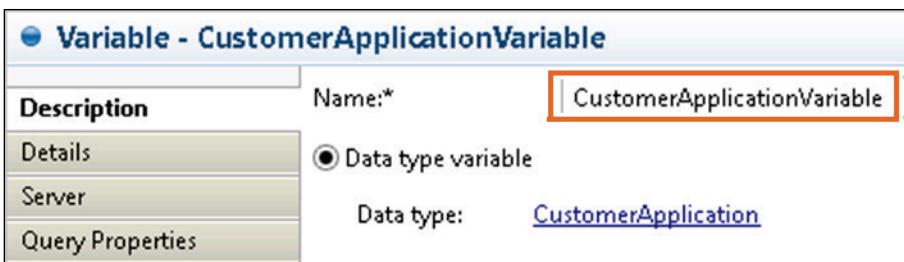
 1. In the **Variables** section, click the plus sign (+) to add a variable.
 2. In the “Add a variable” dialog box, change the **Name** to: `CustomerApplicationVariable2`
 3. In the “Matching data types” section, select the **CustomerApplication** business object.



4. Click **OK**.
5. Save your changes.
4. Change the name of the **Input** variable to CustomerApplicationVariable and change the name of the **Output** variable to MessageVariable.

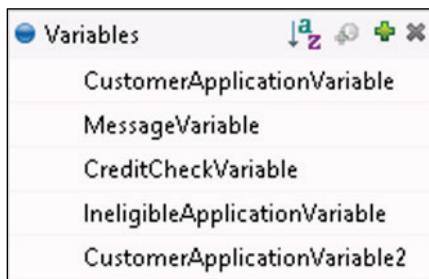
These values correspond to the type of business objects that form the input and output for the process interface.

1. In the **Variables** section of the tray, select the **Input** variable and switch to the **Description** tab in the **Properties** view.
2. Type CustomerApplicationVariable in the **Name** field and press Enter. The **Input** variable is renamed to CustomerApplicationVariable. It is not necessary to refactor it.



3. In the **Variables** section of the tray, select the **Output** variable and switch to the **Description** tab in the **Properties** view.
4. Type MessageVariable in the **Name** field and press Enter. The **Output** variable is renamed to MessageVariable. It is not necessary to refactor it.
5. Save your changes.

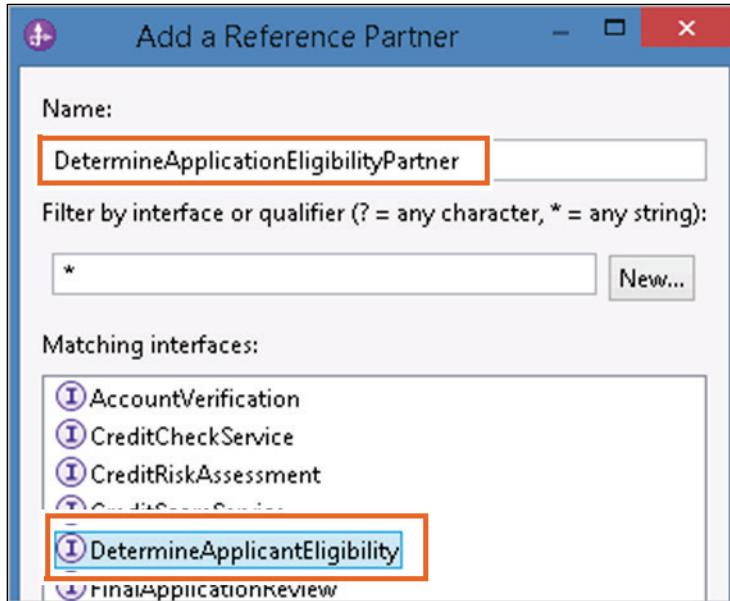
Your process variables look like the following screen capture:



Part 3. Implement WS-BPEL interface partners and reference partners.

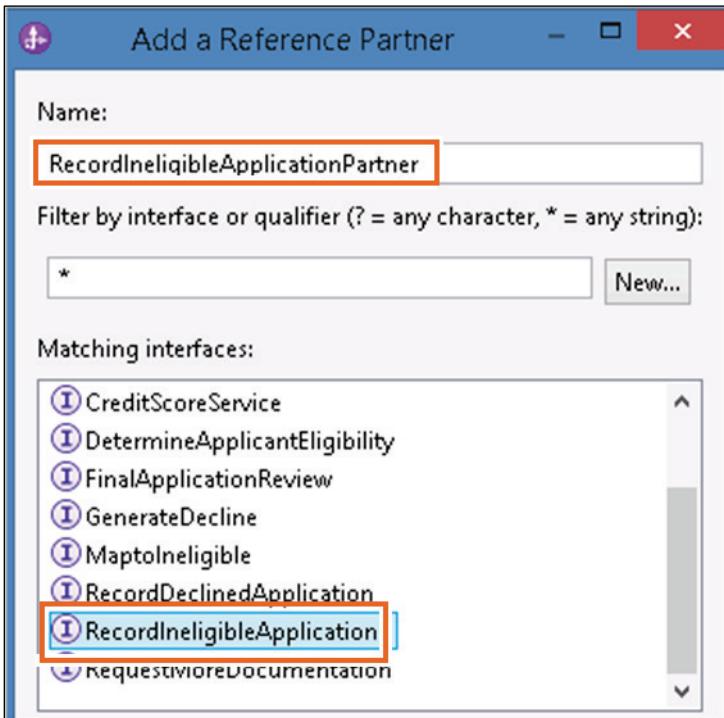
In this portion of the exercise, you add WS-BPEL reference partners to your process. Reference partners represent the service interfaces your WS-BPEL activities call during process execution.

1. Add the **DetermineApplicationEligibilityPartner** reference partner with the **DetermineApplicantEligibility** interface.
 1. In the **Reference Partners** section of the tray, click the plus sign (+) icon to add a reference partner.
 2. In the “Add a Reference Partner” dialog box, change the **Name** to: **DetermineApplicationEligibilityPartner**
 3. In the **Matching interfaces** section, select the **DetermineApplicantEligibility** interface from the list.



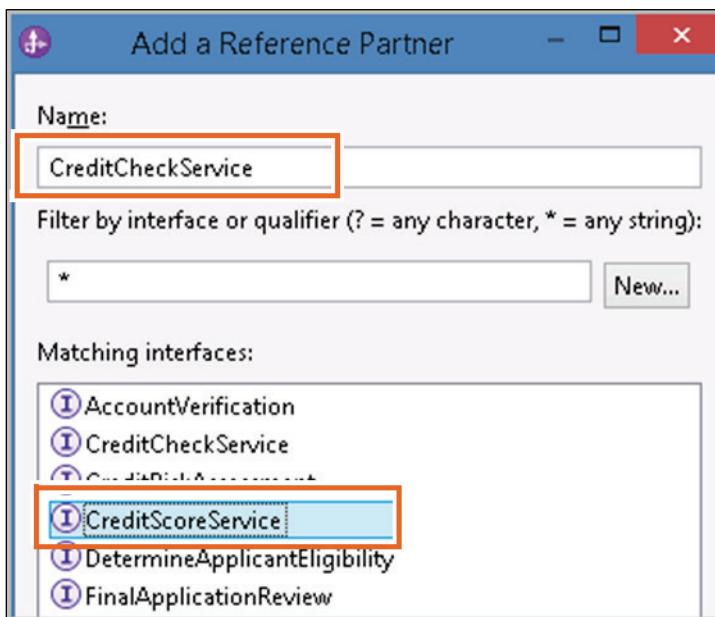
4. Click **OK**.

2. Add the RecordIneligibleApplicationPartner reference partner with the RecordIneligibleApplication interface.
 1. In the **Reference Partners** section of the tray, click the plus sign (+) icon to add a reference partner.
 2. In the “Add a Reference Partner” dialog box, change the **Name** to: RecordIneligibleApplicationPartner
 3. In the **Matching interfaces** section, select the **RecordIneligibleApplication** interface from the list.



4. Click **OK**.
3. Add the CreditCheckServicePartner reference partner with the CreditScoreService interface.
 1. In the **Reference Partners** section, click the plus sign (+) icon to add a reference partner.
 2. In the “Add a Reference Partner” dialog box, change the **Name** to: CreditCheckServicePartner

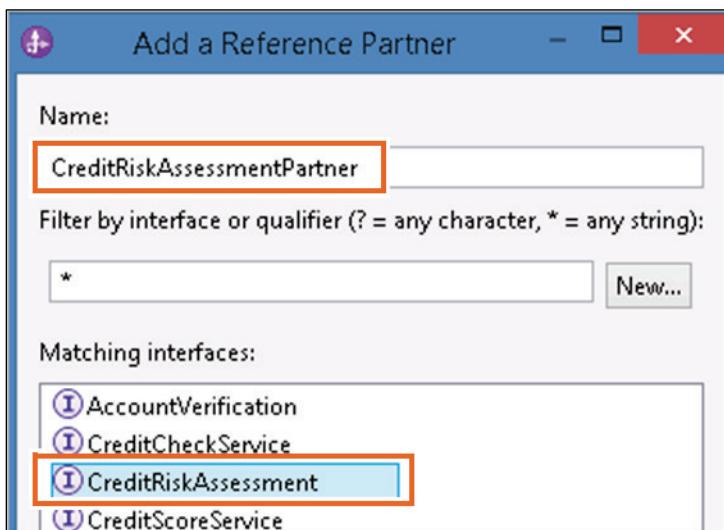
3. In the **Matching interfaces** section, select the **CreditScoreService** interface from the list.



4. Click **OK**.

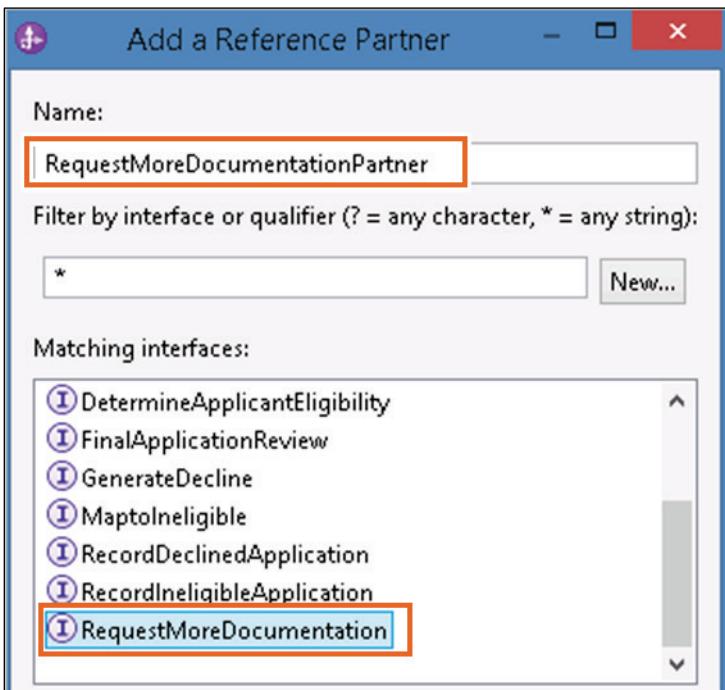
4. Add the **CreditRiskAssessmentPartner** reference partner with the **CreditRiskAssessment** interface.

1. In the **Reference Partners** section, click the plus sign (+) icon to add a reference partner.
2. In the “Add a Reference Partner” dialog box, change the **Name** to: **CreditRiskAssessmentPartner**
3. In the **Matching interfaces** section, select the **CreditRiskAssessment** interface from the list.



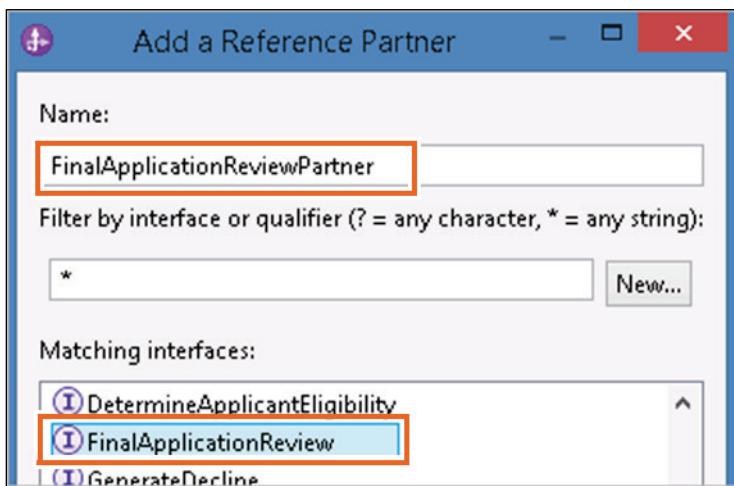
4. Click **OK**.

5. Add the RequestMoreDocumentationPartner reference partner with the RequestMoreDocumentation interface.
 1. In the **Reference Partners** section, click the plus sign (+) icon to add a reference partner.
 2. In the “Add a Reference Partner” dialog box, change the **Name** to: RequestMoreDocumentationPartner
 3. In the **Matching interfaces** section, select the **RequestMoreDocumentation** interface from the list.



4. Click **OK**.
6. Add the FinalApplicationReviewPartner reference partner with the FinalApplicationReview interface.
 1. In the **Reference Partners** section, click the plus sign (+) icon to add a reference partner.
 2. In the “Add a Reference Partner” dialog box, change the **Name** to: FinalApplicationReviewPartner

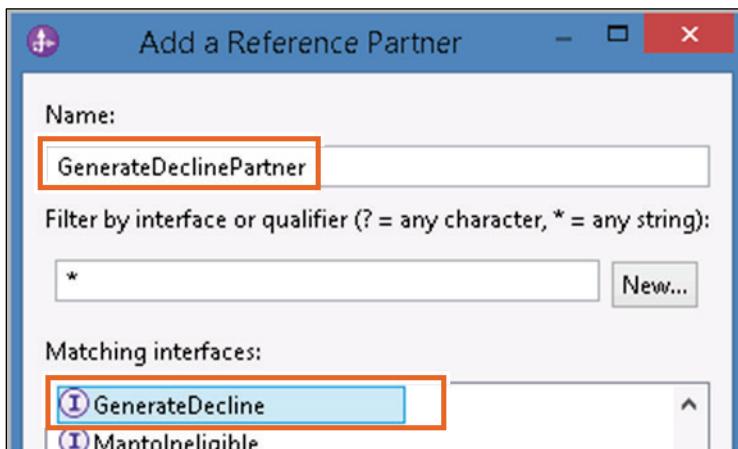
3. In the **Matching interfaces** section, select the **FinalApplicationReview** interface from the list.



4. Click **OK**.

7. Add the **GenerateDeclinePartner** reference partner with the **GenerateDecline** interface.

1. In the **Reference Partners** section, click the plus sign (+) icon to add a reference partner.
2. In the “Add a Reference Partner” dialog box, change the **Name** to: **GenerateDeclinePartner**
3. In the **Matching interfaces** section, select the **GenerateDecline** interface from the list.

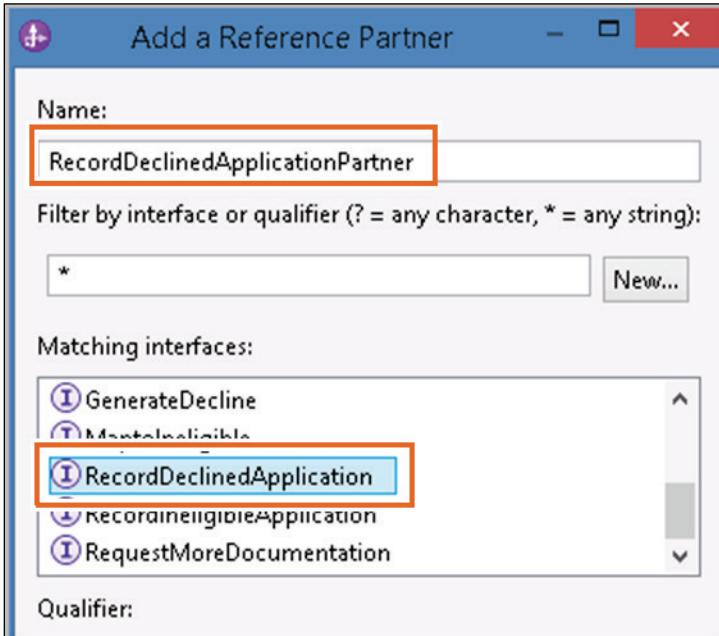


4. Click **OK**.

8. Add the **RecordDeclinedApplicationPartner** reference partner with the **RecordDeclinedApplication** interface.

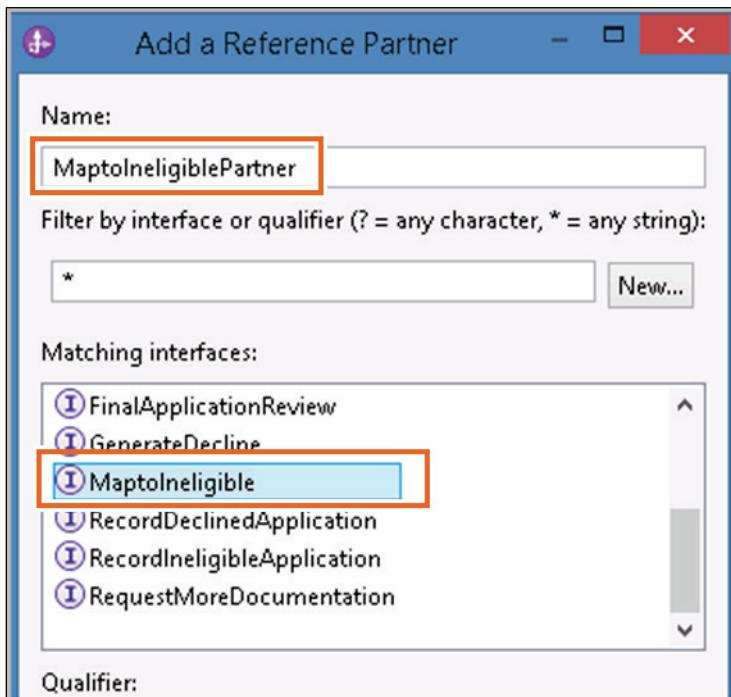
1. In the Reference Partners section, click the plus sign (+) icon to add a reference partner.

2. In the “Add a Reference Partner” dialog box, change the **Name** to: RecordDeclinedApplicationPartner
3. In the **Matching interfaces** section, select the **RecordDeclinedApplication** interface from the list.

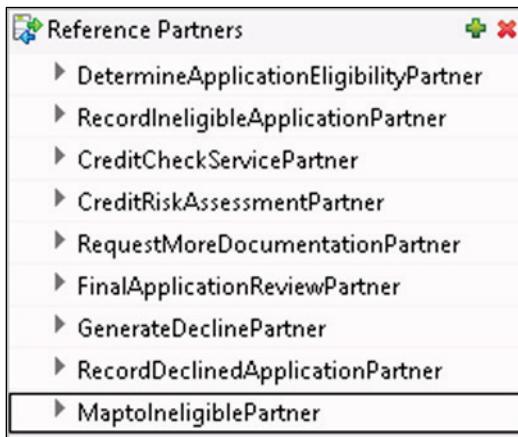


4. Click **OK**.
9. Add the MptoIneligiblePartner reference partner with the MptoIneligible interface.
 1. In the **Reference Partners** section, click the plus sign (+) icon to add a reference partner.
 2. In the “Add a Reference Partner” dialog box, change the **Name** to: MptoIneligiblePartner

3. In the **Matching interfaces** section, select the **Maptolneligible** interface from the list.



4. Click **OK**.
5. Save your changes. Verify that your **Reference Partners** section resembles the following figure:



Question: What is the purpose of these reference partners?

Remember that in a previous statement, it is the expressed intention of this solution to maximize reusability by spreading the implementations of business components through several modules. To facilitate this reusability, you must build “communication conduits” between the modules. These conduits, import, and export components are realized in the implementations as abstract concepts: reference and interface partners.

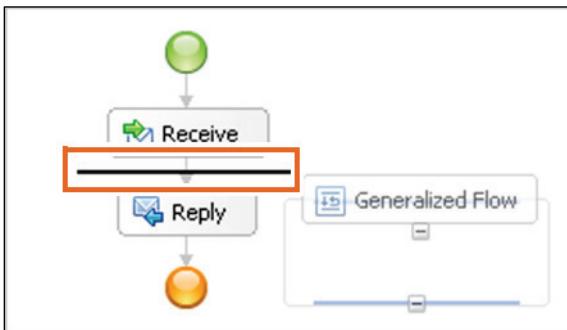
Part 4. Define a business process.

In this portion of the exercise, you create the structured activity (a generalized flow) that acts as the container for the simple and complex BPEL activities you implement in the next exercise.

1. Add a Generalized Flow activity to the process and place the Receive and Reply activities in it.
1. Expand **Structures** in the **Palette** and click **Generalized Flow**.

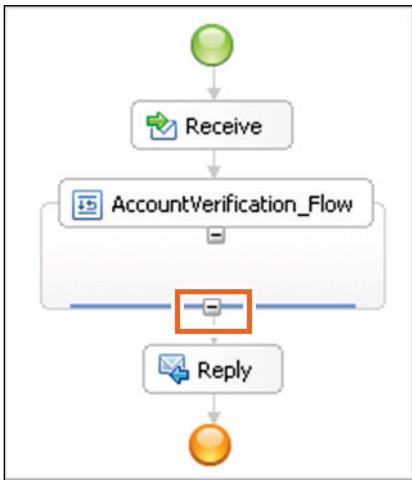


2. Click the space under the **Receive** activity in the editor. A solid black line indicates the insertion point.



3. Ensure that the generalized flow activity is selected, and switch to the **Description** tab in the **Properties** view.
4. Change the **Display Name** from `GeneralizedFlow` to: `AccountVerification_Flow`
For this activity (and all other activities), you alter the Display Name and not the Name. The **Name** field does not support spaces, and the two fields do not have to be the same.

5. Expand the size of **AccountVerification_Flow** by selecting it and dragging the handle at the bottom of the activity.



Note: Structured activities (containing other activities) can be expanded or collapsed either by clicking the plus and minus icons at the top of the activity or by double-clicking the activity.

6. Right-click the process editor and click **Align Parallel Activities Contents Automatically** (to clear it).

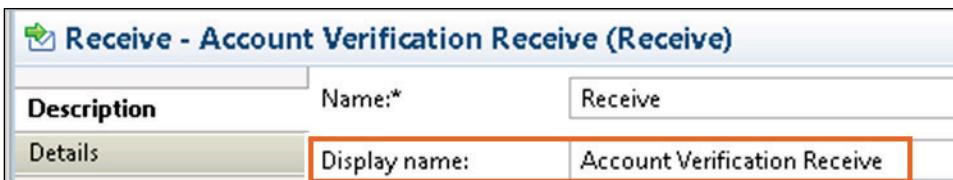
You can manually control the placement of the activities in the diagram by clearing automatic alignment.

7. Save your changes. Ignore any errors in the **Problems** view.

2. Change the Display Name of the Receive activity to:

Account Verification Receive

1. In the editor, right-click the **Receive** activity, and click **Show In > Properties View** from the menu.
2. On the **Description** tab in the **Properties** view, change the **Display Name** to Account Verification Receive and press Enter.



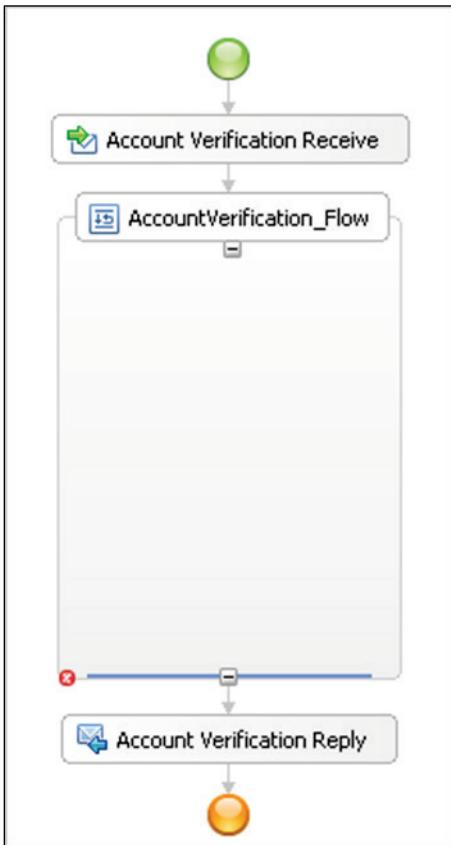
3. Change the Display Name of the Reply activity to: Account Verification Reply

1. In the editor, right-click the **Reply** activity, and click **Show In > Properties View** from the menu.

2. On the **Description** tab in the **Properties** view, change the **Display Name** to Account Verification Reply and press **Enter**.

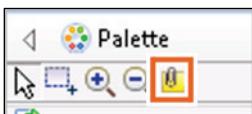


3. Verify that your process looks like the following image:



4. Save your changes. Continue to ignore any errors in the **Problems** view.
4. Add a sticky note to the process with the following text that identifies it as the first version of the process:
AccountVerification process V1

1. In the palette, click the **Note** icon.



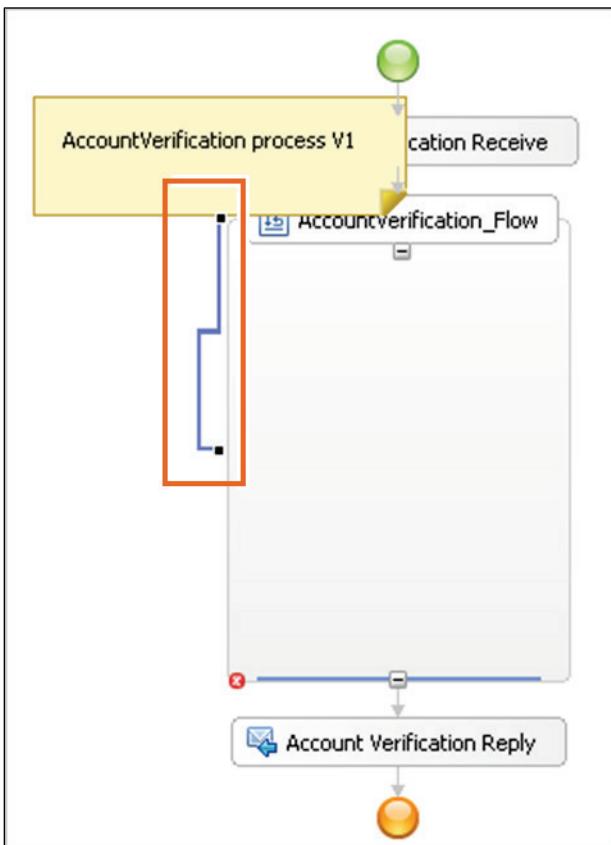
2. Click an empty space in the upper-left corner of the process diagram to add the sticky note.

3. Type the following text in the note: AccountVerification process V1



5. Associate the sticky note with AccountVerification_Flow.

1. Click a blank space on the canvas to escape edit mode in the sticky note.
2. Right-click the note and click **Add Association** from the menu.
3. Click the **AccountVerification_Flow** activity to wire the note to the structured activity.



Note: to aid readability, you can right-click the canvas and toggle the **Hide Notes>Show Notes** option in the menu.

6. Add a snippet to AccountVerification_Flow.

4. In the palette, expand **Basic Actions** and select **Snippet**.
5. Click **AccountVerification_Flow** to add the snippet to the diagram.

6. In the **Properties** view on the **Description** tab, change the **Display name** to: Print Hello World
7. Switch to the **Details** tab in the **Properties** view.
8. Click **Java**. If you are prompted with a dialog box, click **Yes**.
Information: When adding code to a snippet element, you can add the code by using either the visual editor or the text editor, but not both.
9. Copy the "Print Hello World" snippet code from C:\labfiles\Support Files\Ex5\ AccountVerification_Flow_snippet.txt.

```
//*****
// "Print Hello World" snippet
//*****

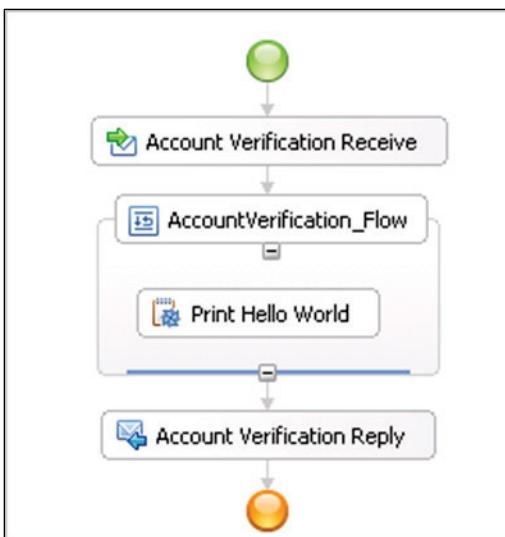
System.out.println("[Java] AccountVerification_Flow - begins");
System.out.println("[Java] Hello World !!!");
System.out.println("[Java] AccountVerification_Flow - ends");
```

10. Paste the snippet in the expression window. Close the text file when you are done pasting the text.

Alternatively, you can manually enter the following text:

```
System.out.println( "[Java] AccountVerification_Flow - begins" );
System.out.println( "[Java] Hello World !!!" );
System.out.println( "[Java] AccountVerification_Flow - ends" );
```

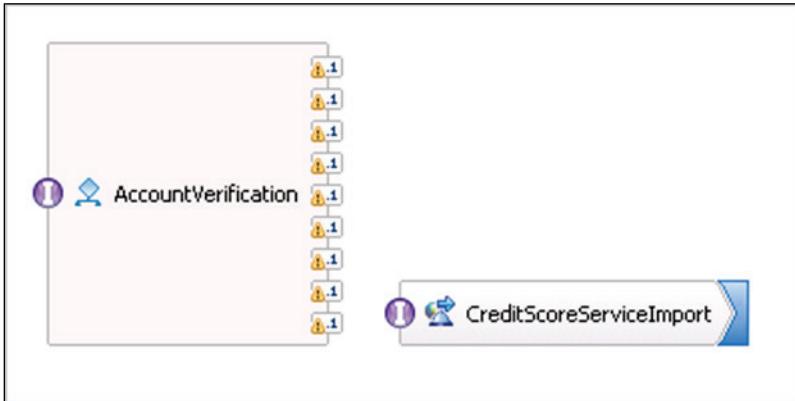
11. To organize the contents for readability, right-click any blank space inside the **AccountVerification_Flow** activity and click **Arrange Parallel Activities Contents Automatically**.



12. Save your changes.

7. Add the AccountVerification service component to the FoundationModule assembly diagram.

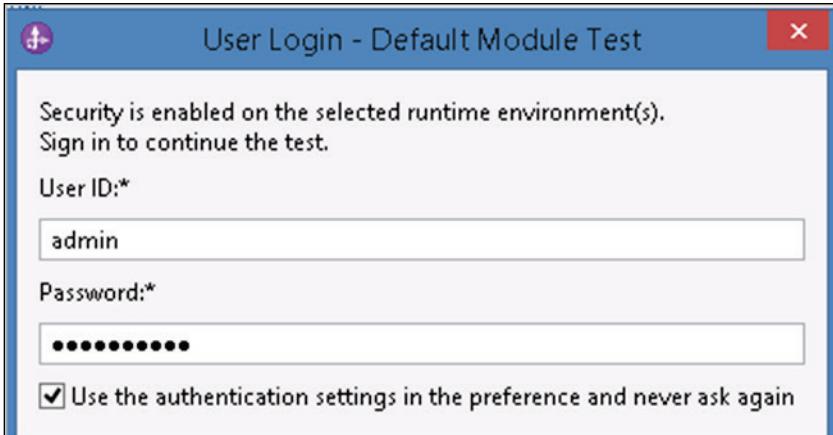
1. In the Business Integration view, expand **FoundationModule**.
2. Double-click **Assembly Diagram** to open the assembly editor.
3. Expand **FoundationModule > Integration Logic > BPEL Processes > AccountVerification**.
4. Drag the **AccountVerification** BPEL process onto the assembly diagram.



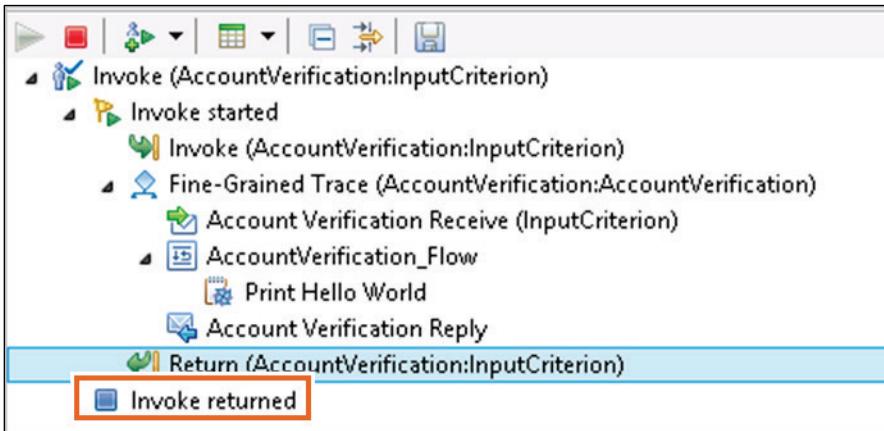
You do not need to wire the services on the assembly editor now. You import the referenced partners and wire them with the AccountVerification process in future exercises.

5. Save your changes.
 8. Test the AccountVerification BPEL process.
 1. If necessary, start the process server by double clicking the desktop shortcut.
 2. In the assembly diagram, right-click the **AccountVerification** BPEL process and click **Test Component** from the menu to open the integration test client. You do not need to input any test data now. Here you are testing the snippet inside the BPEL process that prints the messages in the console.
 3. Click **Continue** on the Events toolbar.
-
4. When you receive the “Select a Deployment Location” dialog box, select **IBM Process Server v8.6 at localhost**, select **Use this location as the default and do not ask again**, and click **Finish**.
- © Copyright IBM Corp. 2010, 2018
- Course materials may not be reproduced in whole or in part without the prior written permission of IBM.
- 8-61

5. When the User Login dialog box is displayed, select the **Use the authentication settings in the preference and never ask again** check box, and click **OK**.



Wait for the test to complete. When the test is finished, you see a stop node that is labeled “Invoke returned” in the Events window.



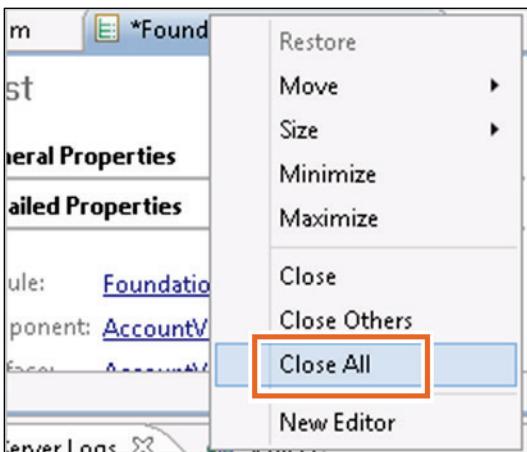
6. You also see messages in the **Server Logs** view that the **AccountVerification** BPEL process returns. Go to the **Server Logs** view and locate the following messages.

[Java] AccountVerification_Flow - begins [Java] Hello World !!

[Java] AccountVerification_Flow - ends

WSVR0220I: Application stopped: FoundationModuleApp
WSVR0200I: Starting application: FoundationModuleApp
WSVR0221I: Application started: FoundationModuleApp
[Java] AccountVerification_Flow - begins
[Java] Hello World !!
[Java] AccountVerification_Flow - ends

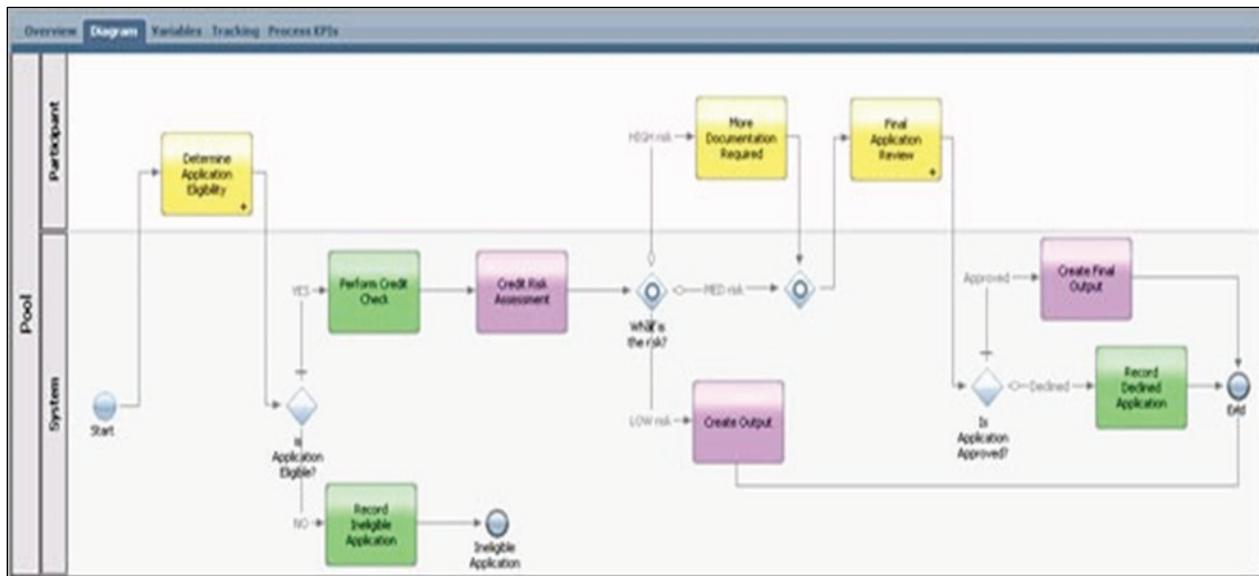
9. Remove the snippet from the AccountVerification BPEL process.
 1. Expand FoundationModule > Integration Logic > BPEL Processes > AccountVerification.
 2. Double-click **AccountVerification**.
 3. In the process editor, right-click the **Print Hello World** snippet and click **Delete** from the menu.
 4. Save your changes.
10. Remove the AccountVerification BPEL process from the FoundationModule assembly editor.
 1. In the Business Integration view, expand **FoundationModule**.
 2. Double-click **Assembly Diagram** to open the assembly editor.
 3. Right-click **AccountVerification** service and click **Delete** from the menu.
 4. In the Confirm Delete dialog box, do *not* select **Also delete the AccountVerification.bpel implementation**. Click **Yes**.
11. Remove the projects 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**. Wait until no messages appear in the IBM Integration Designer status bar.
 3. Click **File > Save All** to save your changes. Continue to ignore any errors in the **Problems** view; they are resolved in later exercises.
 4. Close all open editors in IBM Integration Designer. You can right-click any tab in an open editor and click **Close All**.



5. Close IBM Integration Designer.

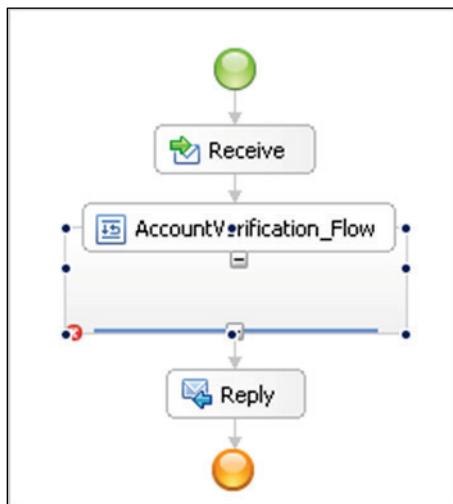
The business process that you created thus far is incomplete. Thus far, you created the basic business process and declared variables, interface partners, and reference partners in preparation for implementing the individual activities of the business process. You add these activities and their implementations in the exercises that follow.

Consider the following process application as your “to be” model.



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

Thus far, you built start and end points only, and the container in which the other activities are going to be built:



You build the remaining activities and their implementations in the exercises that follow.

Results:

In this exercise, you used IBM Integration Designer to create a business process. You also created process variables, interface partners, and reference partners.

Unit 9 Business process basic and structured activities

IBM Training



Business process basic and structured activities

IBM Business Process Manager V8.6

© Copyright IBM Corporation 2018
Course materials may not be reproduced in whole or in part without the written permission of IBM.

Unit objectives

- List and describe the basic activities for business processes
- Define each of the available structured activities for business processes

Business process basic and structured activities

© Copyright IBM Corporation 2018

Unit objectives

This unit describes the various types of activities in BPEL, including basic and structured activities.

Topics

- WS-BPEL basic activities
- WS-BPEL structured activities

Topics

WS-BPEL basic activities

Business process basic and structured activities

© Copyright IBM Corporation 2018

WS-BPEL basic activities

Overview of WS-BPEL activities

- Activities are the individual business tasks that operate with each other to implement the larger business goal, which is represented as the process that contains them
- An activity can be one of several different types: basic activities, structured activities, or activities that are associated with error processing
 - Basic actions are activities that have no structure and do not contain other activities: human task, snippet, and reply
 - Structured activities are activities that contain other activities: sequence and while loop
 - Activities for fault handling and compensation are activities that process expected and unexpected error conditions in processes
- BPEL also uses handlers with certain activities
 - Handlers contain other activities that run based on events or during error processing

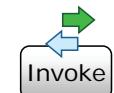
Overview of WS-BPEL activities

As you add activities to a business process in IBM Integration Designer, you can add custom properties to each individual activity. The generated business user clients use the ones that can be queried, and put through a set of APIs.

Custom properties can be used to specify more properties for activities or the process (for example, to associate costs with activities). A custom property has a name and an optional (string) value. The value of custom properties can be specified during authoring time and at run time on a per-instance basis.

Basic activities: Receive, reply, and invoke

- The **receive** activity receives a message sent to a business process
 - Can start a new business process
 - Can restart an existing process
 - Request can be synchronous or asynchronous
- The **reply** activity responds to a message received
 - Typically used as a response to a synchronous request
 - Can return either a response message or a fault message
- The **invoke** activity calls a one-way or a request/response operation that a partner offers
 - Calls another service or business process



Basic activities: Receive, reply, and invoke

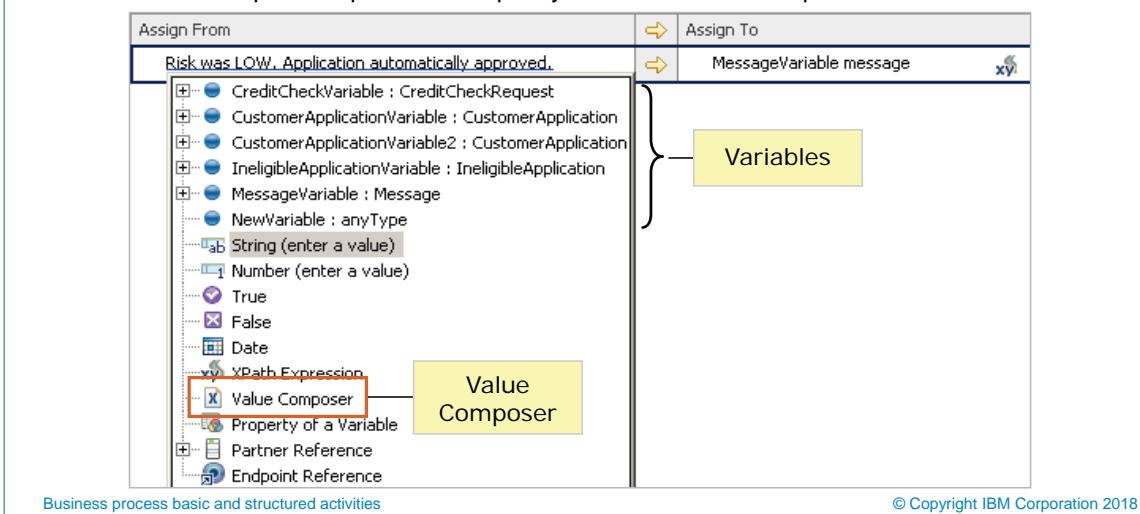
The receive activity is one of the primary means of accepting information into a business process and is typically used at the beginning of the process to accept data and start an instance. The reply activity can be used only with a receive activity and can return a response message (indicating success) or any of various fault messages. Receive and reply can be used in long-running or non-interruptible business processes.

The invoke activity calls services by using partner links. Invocations can be one-way or request/response. It is not necessary to differentiate between synchronous and asynchronous invocations within the business process. The business process engine handles the invocation style for you.

Basic activities: Assign



- An **assign** activity is used to update the values of variables with new data
 - Values can be copied from source variables to destination variables
 - Full XPath support is provided
 - A Value Composer is provided to specify initial values for complex variables



Basic activities: Assign

Assign activities are used with variables. Variables represent the state of the business process and data that is sent to and received from the different services. Variables typically contain business objects.

Different services have different messages that must be populated with data, and the assign activity is the primary way to move information from one variable or message to another. Assign activities do basic mapping of information. For more complex types of mappings, the use of snippets is supported. The BPEL editor consists of a business object map capability, which can be used for mapping specific business objects.

IBM Process Server includes support for XPath V1.0. You can specify an XPath query string to retrieve information from business objects and messages.

Basic activities: Receive choice (1 of 2)

Select an Interface
You can generate a new interface with one operation, or use an existing interface with one, or more operations.

Generate a new Interface
 Select an existing Interface

Interface: Account

Select operations to start the process.

Operations: open close deposit withdraw

Business process basic and structured activities

© Copyright IBM Corporation 2018



- **Receive choice** selects one branch of activities to run based on a receive case
- Receive cases are interface operations

Basic activities: Receive choice

The receive choice activity is a combination of receive and choice. (Receive choice was previously called the pick activity). When a specific message is received, a matching path of activities is run.

Receive choice selects one branch of activities to run based on a receive case.

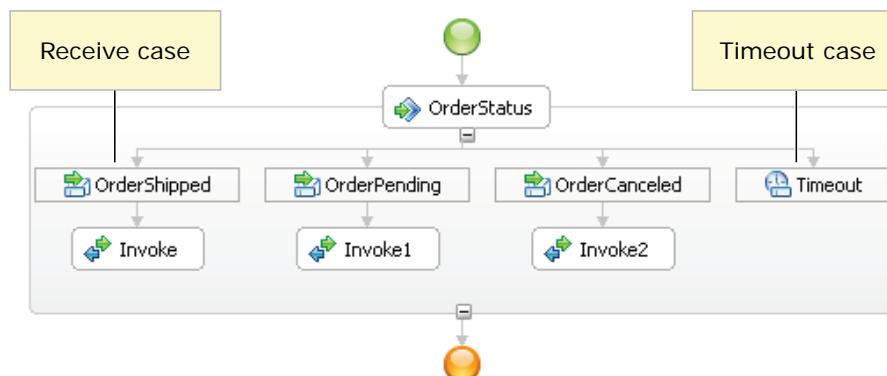
- Receive choice can be used to start a process instance.
- The activity waits for 1 of N possible messages.
- A correlation set is specified for each case.
- Each case can have different permissions.

Receive cases are interface operations.

- If you select an interface with multiple operations when a process is created, it begins with a receive choice instead of a receive.

Basic activities: Receive choice (2 of 2)

- If a message is not received within a certain time, an expression evaluates a timeout case that runs a control path
 - Java and XPath expressions allow a date (visual, Java, or literal date-time) or a duration (visual or Java) value
 - Timeout expressions allow simple calendar, WebSphere CRON, or user-defined values



Business process basic and structured activities

© Copyright IBM Corporation 2018

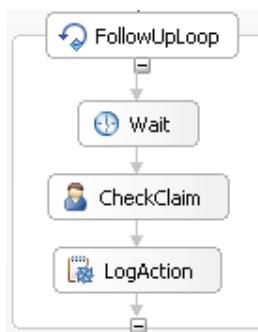
The example that is shown here has three possible paths and a timeout in case none of the expected messages are received during the specific time period. If the order status receive choice activity receives an “order shipped” message, then the invoke activity runs. If an “order pending” message arrives, the invoke1 activity runs. If an “order canceled” message arrives, the invoke2 activity runs. While the receive choice activity is waiting for a specific message to arrive, the entire process instance is in a “sleep mode.” To avoid the process instance from using the server resources indefinitely, timeout logic can be defined. After a certain duration or on a certain date, the process stops waiting for a specific message and then continues.

Timeout is an IBM extension to BPEL. Calendar values are represented in Coordinated Universal Time (UTC). WebSphere CRON is a built-in calendar that uses a list of term expressions that represent elements of time to calculate the interval.

Basic activities: Wait



- A **wait** activity stops the business process for a specific amount of time that an expression evaluates
 - Java and XPath expressions allow a date (visual, Java, or a literal date-time) or duration (visual or Java) value
 - Timeout expressions allow simple calendar, WebSphere CRON, or user-defined values
 - Business calendars are also supported
- Wait is available only for long-running processes



Business process basic and structured activities

© Copyright IBM Corporation 2018

Basic activities: Wait

The wait activity can be used to stop a business process on the current execution path for a specific time. The duration can be either a hardcoded value or calculated dynamically. XPath can be used, or Java can be used for a more complex calculation. Any length of time can be used for a wait activity; therefore, it is available only for use with long-running interruptible processes.

In the example on this page, the FollowUpLoop is used for follow-up handling that an insurance clerk triggers each time that the claim is checked. To repeat the follow-up handling until a decision is made, a while construct is used. When follow-up handling is triggered, the process waits for the time that is specified for follow-up; it is implemented as a wait activity.

The wait activity is available only for long-running (interruptible) business processes. Using the wait activity, you can make the process instance stop for a specific amount of time, or you can terminate the instance by defining a terminate activity.

It is possible to force completion of a BPEL wait activity by using the Business Process Choreographer Explorer.

Basic activities: Human task

- **Human task** activities are an IBM extension to BPEL
 - Human tasks are defined in the BPEL4People specification
- Human task activities in a business process are called inline human tasks
 - Inline tasks send process-related messages to humans for completion
 - Humans can send messages to processes by being given authorization to a receive or a case in a receive choice
- Assigning a task to a human involves the interaction between two editors in IBM Integration Designer
 - You use the process editor to compose a process that requires human interaction
 - You use the human task editor to configure the task

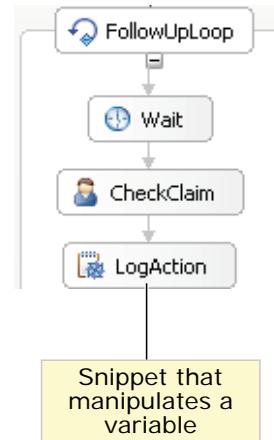


Basic activities: Human task

For more information about human tasks, see Appendix C and Appendix D at the end of this guide.

Basic activities: Snippets

- The **snippet** activity allows execution of Java code in a BPEL process
- Java snippets are an IBM extension to BPEL
- The code runs locally in the BPEL process
 - Snippets in a BPEL process are compiled into a single class
- Snippets are typically used for working with the contents of variables
- Snippets are also used for:
 - Loop counters
 - Data validation
 - Data normalization
- Java snippets can be edited graphically (visual) or textually (Java)
 - A visual snippet can be converted to Java and edited textually
 - Moving back to a visual snippet after converting to Java is not supported



Basic activities: Snippets

The snippet activity is an IBM extension to the BPEL specification. Using snippet activities, you can work with business objects through SDO API calls. You can use an assign to do more robust processing on variables and business process states when the anticipated action cannot be completed. Although the BPEL specification does not support it, the snippet is one of the most useful activities. IBM Integration Designer includes a visual snippet editor, or you can use “plain” Java.

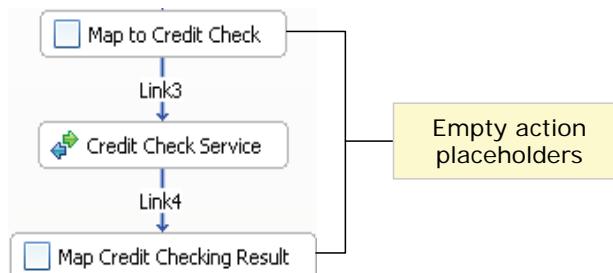
You can use the visual method to create the snippet and click the Java option to view the code that is used to implement that snippet. If you edit code in text mode, you are not able to revert to using the visual tool for that snippet. Snippets that are created in text mode cannot be viewed in visual mode.

A full set of library functions is available for snippets. You can also create your own custom snippets. Snippets can be reused and shared in a team programming environment.

Basic activities: Empty action



- An **empty action** activity acts as a “no-op” instruction in the business process
- The empty action can be used as a synchronization point within the business process
 - To bring together parallel execution paths
- Empty actions can also be used as placeholders
 - For a process activity implemented in the future
 - For development by someone else



Business process basic and structured activities

© Copyright IBM Corporation 2018

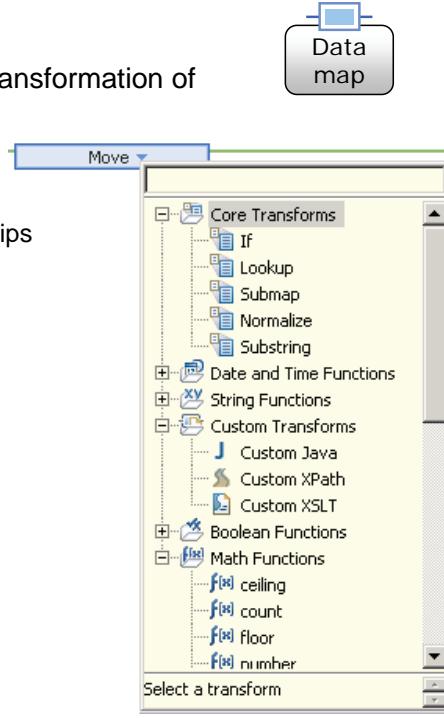
Basic activities: Empty action

Use this activity as an undefined object to act as a placeholder within your process. You might do this activity if you are designing a process that you expect somebody else to implement, or if you are trying to synchronize the activities within a parallel activity.

IBM Training IBM

Basic activities: Data map

- Data maps provide the structural and semantic transformation of business objects
- Maps can be business object maps or XML maps
 - Business object maps are needed only for relationships
- XSL stylesheets that are generated from XML maps are used in XSL Transformation primitives in the mediation flow components
- XML maps can be stored in catalogs for reuse
- Rich set of predefined transformations, including:
 - Core transforms: if, submap, lookup, normalize, and substring
 - Date and time functions
 - String functions
 - Boolean functions
 - Math functions
 - Custom transforms



Business process basic and structured activities © Copyright IBM Corporation 2018

Basic activities: Data map

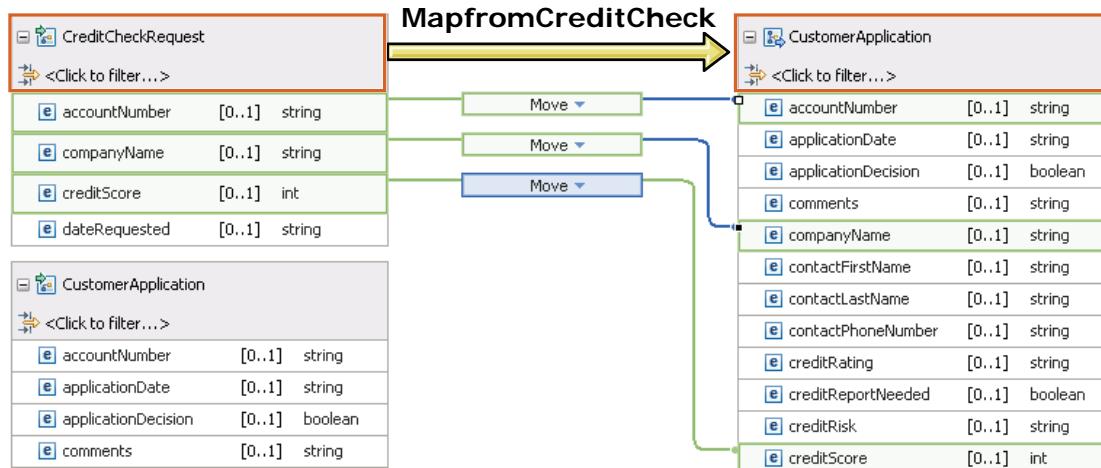
A list of the supported transformations for data maps is included in the course appendixes. For a complete list of built-in transformation functions for XML maps, see the product documentation.

The purpose of a lookup function is to look up a key and return the specific value. This action is done through a comma-separated values (CSV) file, a properties file, or a custom function engine. For the custom function engine, it is easy to create your own .java file. The most frequent example for the lookup functions is the use of a hashtable to look up a key and retrieve another value.

Using the map editor, you can quickly change a map transform into a submap and change from a submap into a local map. The reason for refactoring into a submap is to split up maps into reusable pieces. Large maps can be hard to manage and not reusable. Using submaps makes map content reusable and easier to manage. You can use the “refactor from submap” action to create a local map.

IBM Integration Designer includes a data map catalog. In the catalog, you can filter rows and sort by column headers like name, type, inputs, and outputs. You can also use a text search. By using the toggle buttons in the data map catalog, you can show and hide content tags and namespaces. In addition, you can see other objects that use the map in the references view. Finally, you can create a data map from the data map catalog.

Creating a data map



Business process basic and structured activities

© Copyright IBM Corporation 2018

Creating a data map

To transform data between the source business object and the target business object, use a data map.

To create a data map:

1. In the business integration view, create two business objects.
2. To access the New Business Object wizard, click **File > New > Business Object**. Name these business objects: `BusinessObject1` and `BusinessObject2`
3. Create a BPEL process.
4. In the BPEL process editor, create two variables: `Variable1` and `Variable2`
5. For each variable, click the **Description** tab and in the Data type field, browse to one of the business objects that you created earlier, such that `Variable1` points to `BusinessObject1` and `Variable2` points to `BusinessObject2`.
6. Drop an empty action on the canvas.
7. Click the **Details** tab, and select the data map icon.
8. Select **XML map** or **Business object map** and click **Next**.
9. In the New Data Map wizard, give the new map a name and click **Next**.
10. Select one of the variables as the input and the other one as the output, and click **Finish**.

IBM Training IBM

Common XPath editor

When XPath expressions are used in IBM Integration Designer (as in data map parameter transformations), the graphical XPath Expression Builder is used

The builder provides full XPath support and content assistance

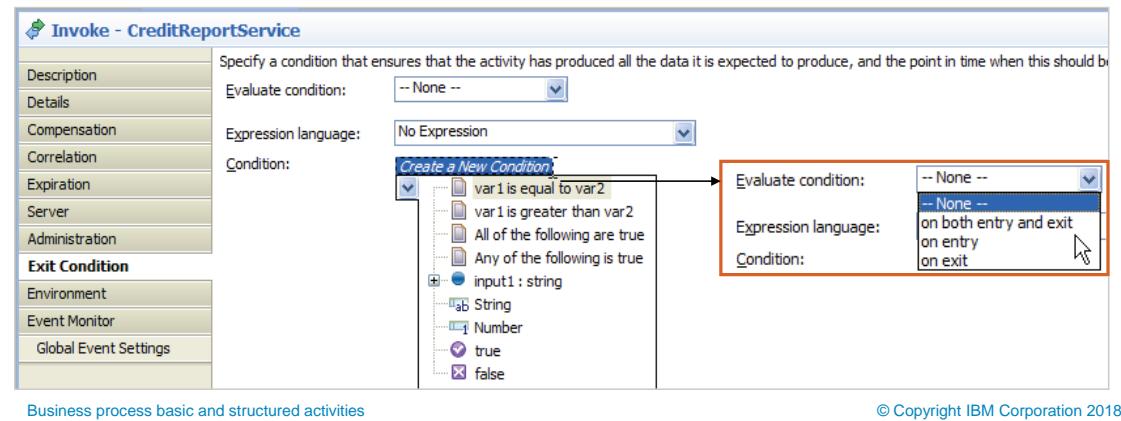
Business process basic and structured activities © Copyright IBM Corporation 2018

Common XPath editor

The common XPath editor is used throughout IBM Integration Designer wherever XPath expressions can be used. The XPath editor can be used for parameter transformations and for mediation flows and BPEL components (such as link conditions and timeout expressions). In the XPath expression builder, you first select a field with the ability to drill down or filter to find elements or business vocabulary aliases. Business vocabulary aliases give you the ability to assign aliases to XPath expressions so that you or your team does not remember complex XPath expressions, just aliases. Then, in the expression builder, you add optional filters or conditions on those elements. You receive immediate validation on the expression.

Activity exit conditions

- Most basic activities and human tasks support exit conditions
 - The **Receive choice** activity does not support exit conditions
- Exit conditions can be set:
 - **On entry:** Determine whether an activity should be done
 - **On exit:** Determine whether navigation continues as expected after completion
 - **On both entry and exit**
- Expressions can be Java, simple (boolean), or XPath



Activity exit conditions

The following process activities support exit conditions: invoke, assign, receive, reply, wait, empty, snippet, data map, compensate, throw, rethrow, terminate, and human task. Exit conditions can be specified:

- **On entry:** The criterion is evaluated immediately after the activity is initiated. If the exit condition evaluates to true, the activity is skipped; otherwise, the activity continues normally.
- **On exit:** The criterion is evaluated before the activity is exited. If the exit condition evaluates to false, the activity is repeated; otherwise, the activity is completed and outgoing connectors are evaluated.
- **On both entry and exit:** The criterion is evaluated immediately after the activity is initiated. If the exit condition evaluates to true, the activity is skipped. If the exit condition evaluates to false at entry, then the activity is run normally. When the activity is complete, the exit condition is evaluated again. If the condition evaluates to false, the activity is repeated; otherwise, the activity is finished and outgoing connectors are evaluated.

WS-BPEL structured activities

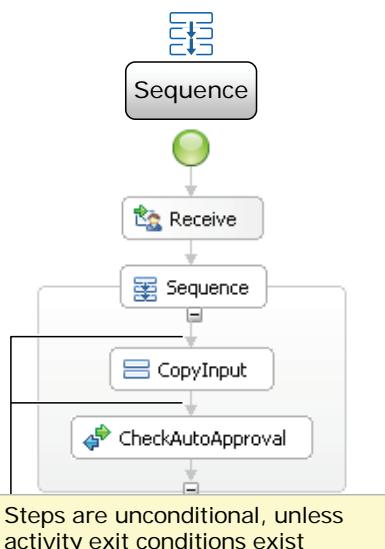
Business process basic and structured activities

© Copyright IBM Corporation 2018

WS-BPEL structured activities

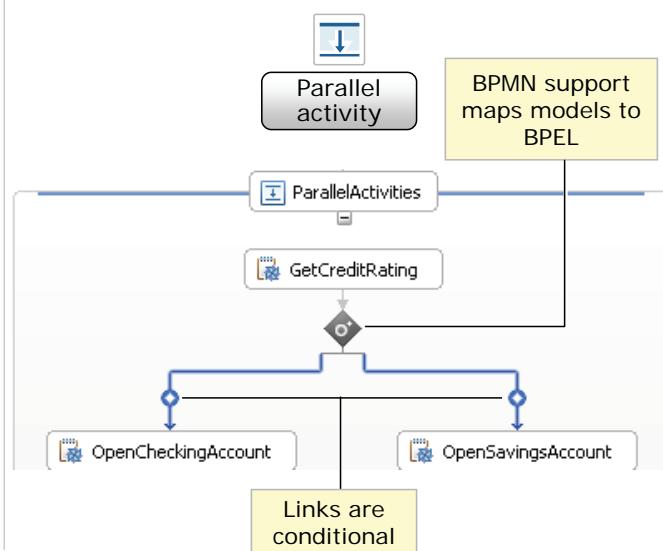
Structured activities: Sequences and parallel activities

- A **sequence** serializes the execution of nested activities
 - Activities run one at a time, in order



Business process basic and structured activities

- A **parallel activity** depicts potentially parallel paths
 - Conditional logic is specified on interactivity links
 - Parallel activities in microflows run sequentially



© Copyright IBM Corporation 2018

Structured activities: Sequences and parallel activities

Parallel activities run branches in parallel only in long-running processes. In a microflow process, all activities run on the same thread, even if they occur within a parallel activity. Parallel activities can use multiple threads in a long-running process. In a parallel activity, links are specified between activities. These links can include conditional logic. This logic can use variable state information from the business process to determine whether execution can proceed.

A sequence runs activities one after another, in order. In a sequence, an activity must complete successfully before the next one runs.

Links and link conditions

- Link conditions evaluate to true, false, or otherwise and are created by using:
 - Java expressions (text or visual)
 - XPath expressions
 - Simple expressions (basic true, false, or otherwise values)
- Multiple true links in structured activities can result in parallel execution paths in a long-running process
 - Parallel paths can be joined with conditions
 - Joins use the logical AND (process waits for execution from all paths), or the logical OR (wait for a single path), to determine when to continue to the next activity
 - If none of the links evaluate to true, a `joinFailure` fault is raised
 - Join failure settings can be used to suppress the fault, skip the activity that threw it, and continue process execution

Links and link conditions

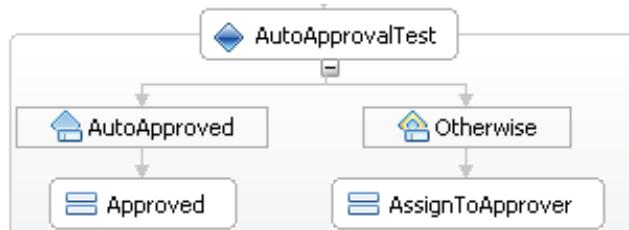
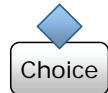
Links can include conditional logic to determine whether processing can proceed on a particular path. Links include support for XPath. Conditions can also be evaluated by using Java programming or a Boolean value. If multiple paths have conditional statements that originate from a single activity, an “otherwise” option can be set and used to run a path. If multiple conditions evaluate to true, parallel execution can be done during a long-running business process.

Join conditions are used to decide whether to wait for execution from all paths or just a single path before continuing a flow.

You can specify a display name for the BPEL link construct. Link names are displayed in the process editor and in the graphical process viewer.

Structured activities: Choice

- A **choice** activity selects one activity branch from a set of case elements that are based on a runtime condition
- The condition is created by using an expression language:
 - Java expressions (visual or text)
 - XPath expressions
 - Simple expressions (boolean true or false)
- At run time:
 - The first case element to evaluate to true is run
 - If no case element evaluates to true, the otherwise element runs
 - If otherwise is omitted, control passes to the activity after the choice activity



Business process basic and structured activities

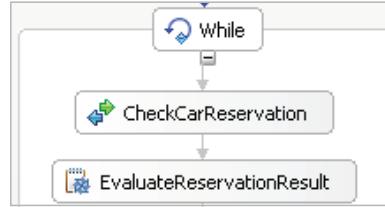
© Copyright IBM Corporation 2018

Structured activities: Choice

The *choice* activity was previously known as the *switch* activity. A choice activity is similar to a case statement. The first condition that evaluates to true determines the processing that takes place. Conditions are checked from left to right, and when one evaluates to true, the activities that are defined under that condition run. An “otherwise” case can also be used to account for situations where none of the conditions evaluate to true.

Structured activities: While loop

- If the condition evaluates to true, the **while loop** repeatedly runs the activities in its scope
- The condition is checked before the first iteration so it is possible that **none** of the nested activities run
- The condition can be created by using:
 - Java expressions (text or visual)
 - XPath expressions
 - Simple expressions (boolean true or false)
- A fault (handled or unhandled) terminates the loop
 - It is more efficient to end the loop by meeting the condition than by throwing an exception



Business process basic and structured activities

© Copyright IBM Corporation 2018

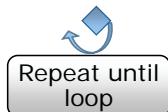
Structured activities: While loop

The *while* activity is used to run activities iteratively. The activities in the scope of the while loop run while a defined condition remains true. The condition is evaluated before activity execution, so it is possible that none of the activities run.

A fault handler can be associated with the while activity to handle failure of any activity in the while loop that is not caught at the while loop construct level. If a fault does occur, any execution in that loop (including parallel activity) halts, and execution continues in the fault handler. Exiting a while loop in this manner is not the best way. You must try to meet the exit condition of the while loop, as more complexity is associated with throwing a fault. The use of faults must be reserved for handling failures in the business logic.

Structured activities: Repeat until loop

- The **repeat until loop** executes the activities in its scope at least one time and then continues to loop until the condition evaluates to true
- The condition is checked at the *end* of each iteration so the activities run **at least** one time
- The condition can be created by using:
 - Java expressions (visual or text)
 - XPath expressions
 - Simple expressions (boolean true or false)
- A fault (handled or unhandled) terminates the loop
 - It is more efficient to end the loop by meeting the condition than by throwing an exception



Business process basic and structured activities

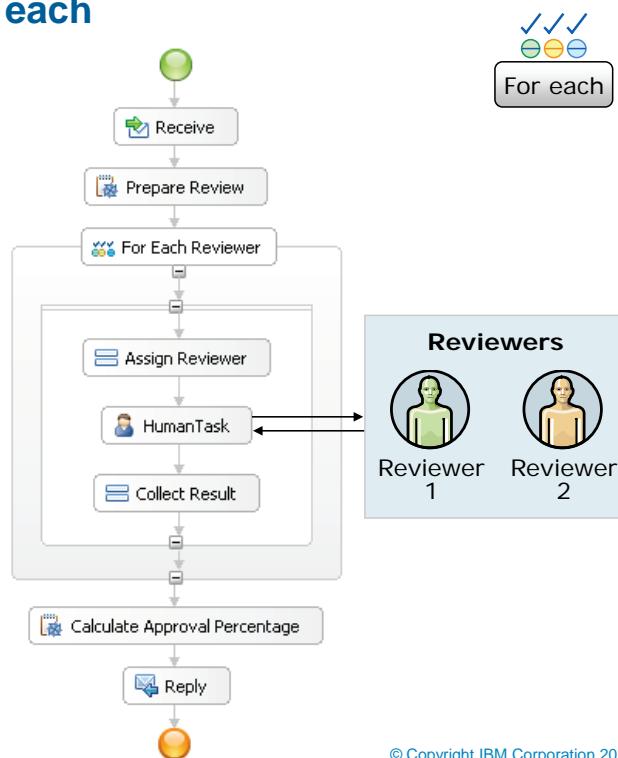
© Copyright IBM Corporation 2018

Structured activities: Repeat until loop

The *repeat until* activity provides for repeated execution of contained activities. The activities run until the condition evaluates to true. The condition is tested after each execution of the body of the loop, so the contained activities run at least once.

Structured activities: For each

- The **for each** activity enables the bundling of work
 - Requests that require a dynamic number of reviewers
 - Continue after some of the quotes are received
- Within the bundle, you control:
 - Whether a dynamic number of branches can be run serially or in parallel
 - Whether all branches are required for completion
 - Early exit criterion that specifies termination after a certain subset of branches



Business process basic and structured activities

© Copyright IBM Corporation 2018

Structured activities: For each

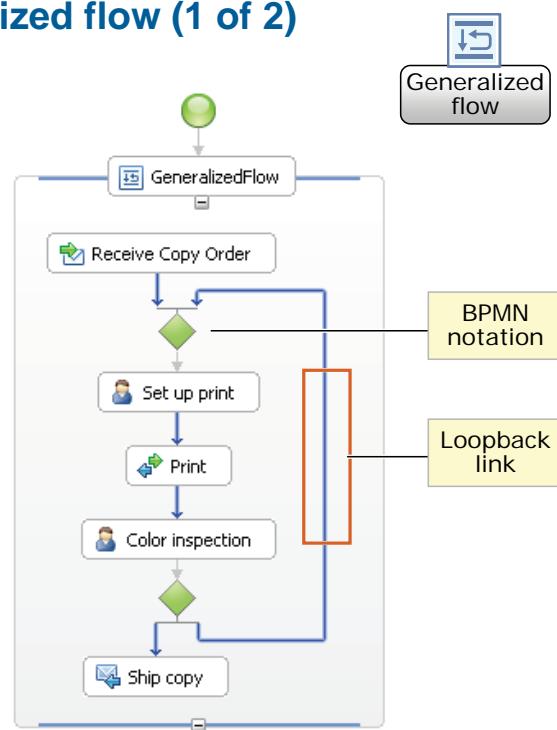
For each is useful in cases where sets of independent data are processed or where independent communication with different partners can be done in parallel. Unlike the parallel activity, the number of parallel branches in a “*for each*” is not known when the process is modeled. Therefore, a “*for each*” behaves like a flow with “n” similar child activities in which the links are constrained. A specified counter variable is used to iterate through a number of parallel branches that a start value and a final value control.

Exit criteria can be specified when the process does not require the completion of all branches (when enough recipients respond or when other reviews are not required because one reviewer rejects the document), for example:

- The author of a document starts a review process human task.
- The reviewers are unknown at authoring time. They are determined during process start through the input message.
- The document review happens in parallel.
- When enough reviews are complete, the “*for each*” is exited.

Structured activities: Generalized flow (1 of 2)

- Generalized flows:
 - Allow easy “backward” links in the process (that go back to a previous step): A loop back condition controls when the “back flow” occurs
 - Can use “catch” or “catch all” fault links instead of fault handlers
 - Support parallel execution paths
- Generalized flows support important modeling scenarios
 - Model arbitrary loop logic (return to a previous activity)
 - Frequently used with human tasks
 - Example: A claims process where you must arbitrarily jump back to redo previous activities



Business process basic and structured activities

© Copyright IBM Corporation 2018

Structured activities: Generalized flow

Generalized flows are an IBM extension to BPEL. In prior versions, generalized flows were termed cyclic flows.

In this example, the order of a color copy comes in through the “receive copy order” activity. A human task is then assigned to an employee in the “set up print” activity to set up the source file for printing. Part of this setup process includes adjusting the colors so that they are a close match to the original. When this setup process is complete, the flow continues to the “print” activity, which produces the copy. The output is then directed to another human task, the “color inspection” activity. This employee is responsible for deciding whether the colors are a match to the original. If they are not, then the evaluation is set to “false” and the flow is returned to the “set up print” activity. This loop continues until the employee at the “color inspection” activity decides that the color is acceptable, and the process is allowed to progress to the “ship copy” activity.

The generalized flow activity is similar to a parallel activity in that you can nest other process activities within it, and then control the execution order of those activities through links. The generalized flow and parallel activities can also be used to model a business process in a graph-oriented manner. The main difference between the two activities is that the generalized flow supports cycles (conditional links to loop back to previous activities in the sequence) and fault links. The parallel activity does not.

When faults occur in your business process, fault handlers are typically engaged to deal with the fault. The generalized flow activity offers a simplified “fault handling” procedure. From any scope or basic activity (excluding the throw and rethrow activities), you can add one or more “fault links.” If a specified fault occurs while an activity is running, the fault link in the activity is followed. You can define “catch” fault links for various conditions, or you can create a “catch-all” fault link, which is followed when any fault occurs that a “catch” fault link does not cover. If multiple fault links are modeled for the same activity, “best match” decides which fault link to follow. The “fault catching” rules are the same as for fault handlers. No more than one fault link can catch a fault.

You can terminate the fault link at any activity within the generalized flow. For example, you can choose to direct the fault link to a terminate activity to have the business process stop if a fault occurs. You might also want the business process to skip the rest of the steps in the generalized flow and exit to the next activity. In this case, you would terminate the fault link at the last activity in the generalized flow. You can also use a fault link to create a cycle and loop back to a previous activity.

Structured activities: Generalized flow (2 of 2)

- In a generalized flow, gateways are used to define the diverging and converging behavior when more than one link is the target or source of an activity
 - Split/Merge, Fork/Join, and Inclusive OR
- Gateways use Business Process Model and Notation (BPMN) to support IBM Process Designer notation

The diagram illustrates the configuration of BPMN gateways in IBM Process Designer. On the left, three gateway types are shown with their corresponding symbols: Split/Merge (diamond), Fork/Join (plus sign), and Inclusive OR (diamond with a circle inside). To the right, two configuration panels are displayed:

- Outgoing Gateway - Determine Credit Rating**: This panel shows the "Navigation" tab selected. It includes a section titled "Specify the navigation behavior for the outgoing links" with three options: Split - The process navigates only the first outgoing link whose transition condition is true, Fork - The process navigates all of the outgoing links, and Inclusive OR - The process navigates all outgoing links whose transition condition is true.
- Incoming Gateway - Determine Credit Rating**: This panel shows the "Navigation" tab selected. It includes a section titled "Specify the navigation behavior for the incoming links" with three options: Merge - The process waits only for the first incoming link, Join - The process waits for all of the incoming links, and Inclusive OR - The process waits for all of the navigated incoming links.

Business process basic and structured activities © Copyright IBM Corporation 2018

Business Process Model and Notation (BPMN) is a standardized graphical notation for creating diagrams of business processes. You can learn more about BPMN at: <http://www.bpmn.org/>

The visual representation of “graph constructs” in BPEL (parallel activities) means that for each activity with more than one incoming or outgoing link, a (BPMN-like) diamond (or gateway) represents the incoming and outgoing navigation behavior.

Structured activities: Scope

- A **scope** is a behavioral container for one or more activities in your process
 - Encapsulates variables and correlation sets (state)
- By definition, your entire process is contained within a single global scope
- You can nest other scopes within the global scope, forming a hierarchy
- Each scope can have its own:
 - Fault handlers (for expected error processing in the scope)
 - Event handlers (for event processing in the scope)
 - Compensation handlers (for unexpected error processing in the scope)
 - Local variables
- Scopes support dynamic runtime behavior



Structured activities: Scope

Fault handlers, compensation handlers, and event handlers are all associated with a particular scope. Scopes are supported as defined in the BPEL specification. Scopes encapsulate correlation sets and the state of a business process to make particular values available to a specific set of activities. Scopes can be established to define event handlers and compensation handlers.

Scopes can also be nested, allowing variables with the same name to exist in the business process as a unique instance. If a variable is defined at a particular scope, that variable is accessed when called by name. Variables that are defined at a parent scope are available while the name is different. When the scope ends, the variable is no longer available to downstream activities.

Structured activities: Collaboration scope (1 of 2)

- **Collaboration scope** is the preferred tool for the case paradigm
- A “case” is the product of a workflow or part of a workflow
- A case can be any number of things, including:
 - Evaluation of a job application
 - Ruling on an insurance claim
- Any business process can be handled as a case, but it is ideally suited to situations where the task owner uses knowledge and experience to:
 - **Expedite the process:** Close a case without going to trial
 - **Trigger subprocesses:** A doctor orders an extra blood test
 - **Repeat a number of activities:** A second interview
 - **Take an alternative path through the main flow of activities:**
An insurance adjuster sees a claimant for added details
- The case model creates workflows that are well-defined but give the staff the flexibility to use their skills and judgment to adapt the flow to business needs
 - These business processes are also called “enhanced dynamic workflows”

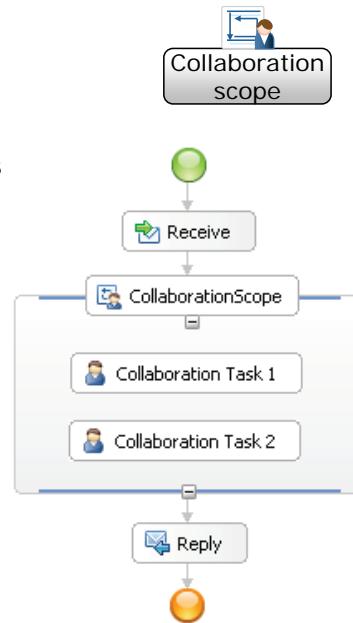


Structured activities: Collaboration scope

The collaboration scope activity allows enhanced dynamicity for knowledge workers by supporting case handling systems. Case handling systems assist knowledge workers in processing a business case (rather than constraining them). In a “case,” the high-level process is well-defined; however, many exceptions cannot be efficiently predefined. The case does not necessarily run in a straightforward manner, but might require you to jump (redo or skip steps).

Structured activities: Collaboration scope (2 of 2)

- Case handling support is provided by combining dynamicity features and the collaboration scope activity
 - BPEL Process Choreographer Explorer and Business Space support process dynamicity at run time
- You can create business logic within your collaboration scope by adding basic activities
 - You cannot include structured activities in a collaboration scope
- Exit conditions can be used to automatically skip and repeat steps
- You set the administrators for the collaboration scope to specify the people who do manual skips and jumps at run time
 - The default setting is everybody



Business process basic and structured activities

© Copyright IBM Corporation 2018

Use Collaboration scopes to create enhanced dynamic workflows: business processes in which the business logic can be adapted at run time. For example, the assigned worker can decide to repeat an activity, start a subtask, or skip some steps in the business process.

You can configure a scope activity so that users who interact with a runtime instance of the process have administrative authority over the activities that are nested within the scope activity. You can achieve enhanced dynamic behavior more directly by using a “collaboration scope” in which the associated administration task is automatically generated when the collaboration scope is added to the business process.

In the runtime environment, the business process can stop at an activity that is nested within a collaboration scope or a scope for which dynamicity is configured. In that case, an authorized individual can skip, undo, or redo that nested activity. These options are available through a client such as the Business Process Choreographer Explorer. The ability to dynamically modify a process that is already deployed to a runtime environment is especially useful in cases where the process describes a series of steps that are not always necessary.

Unit summary

- List and describe the basic activities for business processes
- Define each of the available structured activities for business processes

Checkpoint questions

1. What is the value of having a generalized flow activity?
2. What is the difference between the parallel activity and the sequence activity?
3. List and define three basic process activities.
4. What type of process paradigm is the collaboration scope designed to support?
5. What is the primary purpose of the “for each” activity?
6. How does a choice activity function?

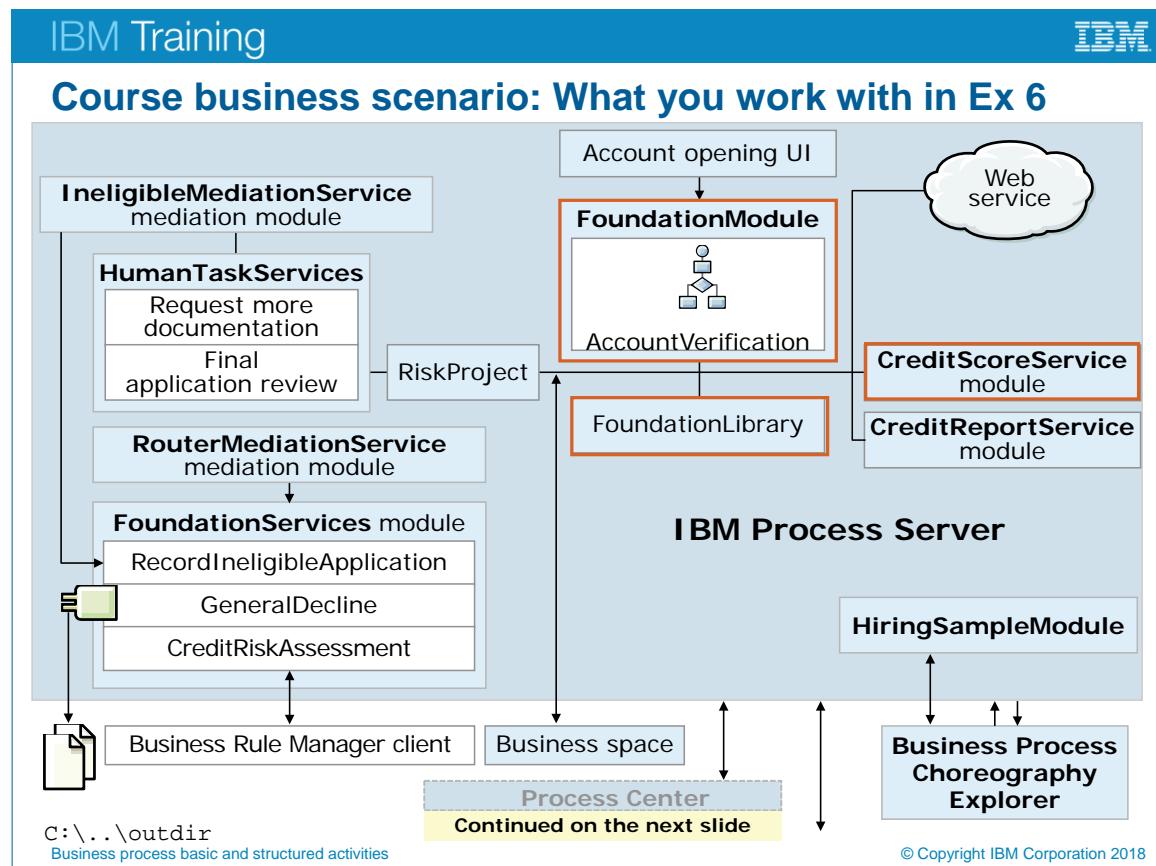
Checkpoint answers

1. The generalized flow activity can be used for conditional links to loop back to previous activities in the process.
2. Parallel activities can potentially run activity paths in parallel in interruptible (long-running) processes. Sequences consist of a serialized execution of activities that occur one right after another.
3. Any of the following activities:
 - receive (receive message)
 - reply (respond)
 - invoke (call partner)
 - assign (update variable)
 - wait (pause execution)
 - empty (no-op)
 - empty snippet (Java code)
 - human task (create work items for people)
 - data map (transform business object)
 - receive choice (choose branch, which is based on receive case)
4. The “case” paradigm.
5. The “for each” activity enables the bundling of work.
6. A choice activity selects one activity branch from a set of case elements based on a runtime condition.

Exercise 6: Creating a business process, part II

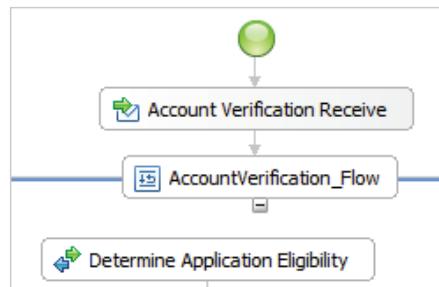
After completing this exercise, you should be able to:

- Implement basic BPEL activities in a business process
- Implement structured activities in a business process
- Compare the BPEL to the BPD in IBM Process Designer



Course business scenario: What you work with in Ex 6

Complete Account verification process (1 of 4)

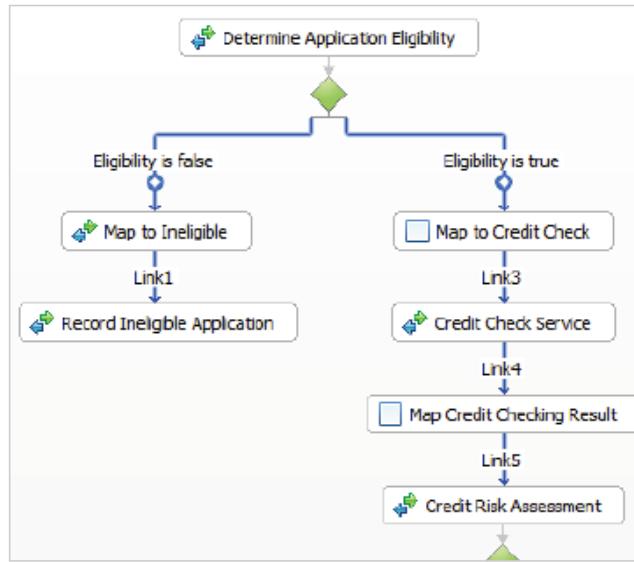


Business process basic and structured activities

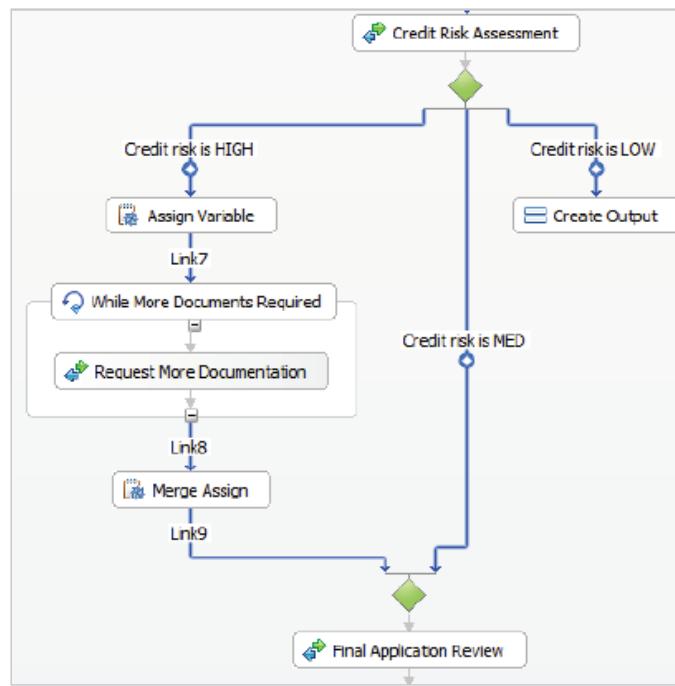
© Copyright IBM Corporation 2018

Complete Account verification process

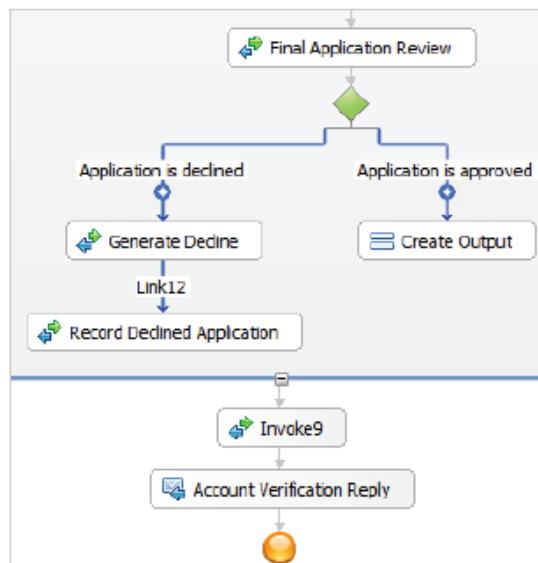
Complete Account verification process (2 of 4)



Complete Account verification process (3 of 4)



Complete Account verification process (4 of 4)



Components that are required for Exercise 6 (1 of 2)

Prebuilt components that are imported in the lab:

- 1. FoundationModule**
- 2. CreditScoreService**
- 3. FoundationLibrary**
- 4. AccountVerification**

- BPEL process that you started in Exercise 6
- You complete building the process in this exercise

New components that you create in the lab:

- 1. AccountVerification**

- BPEL process that you started in Exercise 6
- You complete building the process in this exercise

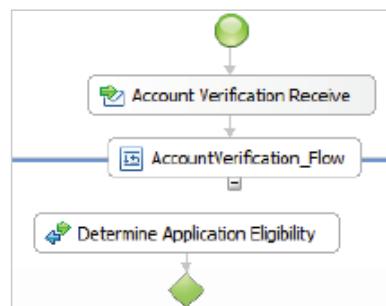
Note:

The next slide lists all of the subcomponents of the **AccountVerification** process that are built in this lab.

Components that are required for Exercise 6

Components that are required for Exercise 6 (2 of 2)

- Activities that are created in the AccountVerification process:
 - Invoke activity
 - Determine Application Eligibility
 - Map to Ineligible
 - Record Ineligible Application
 - Credit Check Service
 - Credit Risk Assessment
 - Request More Documentation
 - Final Application Review
 - Generate Decline
 - Empty action activity
 - Map to Credit Check
 - Map Credit Checking Result
 - Assign and loop activity
 - Create Output assign activity
 - While More Documents Required while loop activity

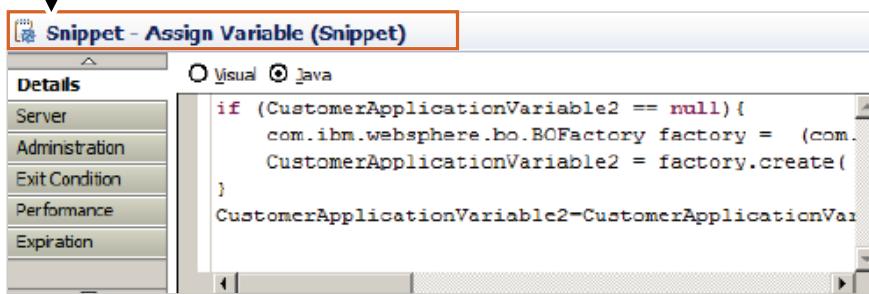


1. You create the first activity: **Determine Application Eligibility** from where the **AccountVerification** process flow starts. The **Determine Application Eligibility** activity examines the customer application to determine whether the customer is eligible for an account. It invokes the **InputCriterion** operation of the **DetermineApplicantEligibility** interface of **DetermineApplicationEligibilityPartner**.
2. The **Map to Ineligible** invoke activity invokes the **InputCriterion** operation of the **MaptoIneligible** interface of **MaptoIneligiblePartner**.
3. The **Record Ineligible Application** activity archives any customer application that is determined to be ineligible. You implement the application that archives the application in a later exercise.
4. The **Map to Credit Check** transforms the data from a customer application business object into a format suitable for the credit score service to understand.
5. The **Credit Check Service** activity invokes the **calculateCreditScore** operation of the **CreditScoreService** interface of **CreditCheckServicePartner**.
6. After the credit score is returned, the data is again transformed. In this case, **Map Credit Checking Result** transforms it back into a customer application business object.

7. The **Credit Risk Assessment** activity examines the credit score that is received and does one of the following three actions:
 - If the credit risk is high, more documentation is requested before final review.
 - If the credit risk is medium, final employee review is requested before approval.
 - If the credit risk is low, the application is approved.
8. The **While More Documents Required** activity continues to request documentation while the comment attribute is equal to None. When an employee reviews the application, the comment attribute is populated and the loop terminates.
9. The **Request More Documentation** activity invokes the **InputCriterion** operation of the **RequestMoreDocumentation** interface of **RequestMoreDocumentationPartner**.
10. The **Final Application Review** activity invokes the **InputCriterion** operation of the **FinalApplicationReview** interface of **FinalApplicationReviewPartner**. This activity invokes a human task that allows an employee to review an application.
11. The **Generate Decline** activity invokes the **InputCriterion** operation of the **GenerateDecline** interface of **GenerateDeclinePartner**.
12. The **Record Declined Application** activity invokes the **InputCriterion** operation of the **RecordDeclinedApplication** interface of **RecordDeclinedApplicationPartner**.
13. The **Create Output** activity sets the process output message to: “Application was approved.” The process returns the message when creditRisk is HIGH or MED and the person who reviews the application approves it. This **Create Output** activity sets the process output message to: “Risk was LOW. Application automatically approved.” The process returns the message when creditRisk is LOW and the application flows directly from Credit Risk Assessment to Account Verification Reply.

Code snippets that are created in the AccountVerification process in Exercise 6

- ⑩ Determine App Eligibility -> Map to Ineligible code snippet
- ⑩ Determine App Eligibility -> Map to Credit Check code snippet
- ⑩ Assign Variable code snippet**
- ⑩ Credit Risk Assessment -> Assign Variable code snippet
- ⑩ Merge Assign code snippet
- ⑩ Credit Risk Assessment -> Final Application Review code snippet
- ⑩ Final Application Review -> Generate Decline code snippet
- ⑩ Final Application Review -> Create Output code snippet
- ⑩ Credit Risk Assessment -> Create Output code snippet



Business process basic and structured activities

© Copyright IBM Corporation 2018

Code snippets that are created in the AccountVerification process in Exercise 6

Exercise 6: Creating a business process, part II

Purpose:

In this exercise, you use IBM Integration Designer to implement a complex WS-BPEL business process. You use a combination of simple and structured activities to choreograph the service invocations that are designed to implement the account verification process.

The Web Services Business Process Execution Language (WS-BPEL) provides syntax for specifying the behavior of a business process in a platform-independent manner. It is used to coordinate a series of service invocations to fulfill a business task. Although the language provides conditional and control logic, most of the work is done through the invoked services.

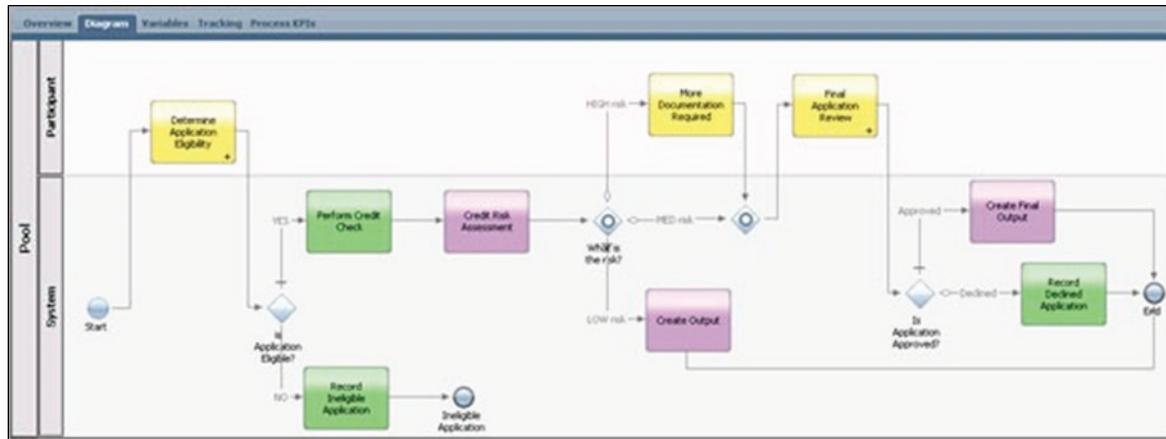
Combining WS-BPEL with the SCA programming model allows for the coordination of SCA services into much larger units of work. Individual SCA services can be brought together and can benefit from the advanced capabilities of WS-BPEL.

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.

In this exercise, you use the WS-BPEL graphical editor to define the core business logic for the “account verification” process. You implement the simple and structured activities that choreograph the service invocations that the process requires.

Compare with the model from IBM Process Designer. 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. Implement basic and structured BPEL activities in a business process.

1. Implement the AccountVerification business process

1. On your desktop, open the **Exercise Shortcuts** folder.
2. Double-click the shortcut that is labeled **Exercise 6**. 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.

Note: Because the business process you began developing in Exercise 6 is incomplete, you see errors in the Problems view after you open the workspace. These errors are resolved when you complete this exercise.

2. Open the AccountVerification business process.

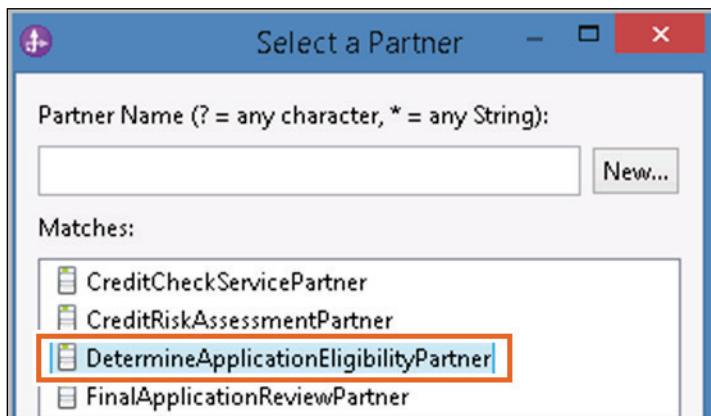
1. In the Business Integration view, expand FoundationModule > Integration Logic > BPEL Processes > AccountVerification.
2. Double-click **AccountVerification** to open the process editor.

3. Add an invoke activity with the Display Name: Determine Application Eligibility

The **Determine Application Eligibility** activity invokes the **InputCriterion** operation of the **DetermineApplicantEligibility** interface of **DetermineApplicationEligibilityPartner**.

The **Determine Application Eligibility** activity examines the customer application to determine whether the customer is eligible for an account. Eligibility is predetermined for the test data by using a Java component that you implement later. In practice, this activity would invoke a service such as a human task.

1. In the palette, expand **Basic Actions** and click **Invoke**.
2. Click inside **AccountVerification_Flow** to place the activity on the diagram.
3. In the **Properties** view on the **Description** tab, change the **Display Name** to: Determine Application Eligibility
You do not need to change the value in the **Name** field. The **Name** field does not support spaces and is not required to match the value in the **Display Name** field.
Note: All of the activity and link display names are listed in C:\labfiles\Support Files\Ex6\ ActivityNames.txt. You can use this file to copy and paste the text instead of typing it.
4. Switch to the **Details** tab in the **Properties** view.
5. Beside the **Partner** field, click **Browse** and select **DetermineApplicationEligibilityPartner** from the “Select a Partner” dialog box.



6. Click **OK**.

7. Verify that the **Interface** field is set to `DetermineApplicantEligibility`, the **Operation** field is set to `InputCriterion`, and the **Use data type variables mapping** option is selected.

Invoke - Determine Application Eligibility (Invoke)	
Description	Partner:*
Details	Interface:*
Server	DetermineApplicationEligibilityPartner
Administration	Operation:*
Exit Condition	<input checked="" type="checkbox"/> Use data type variables mapping

In the **Inputs Read from Variable** column, click the **none** link and select **CustomerApplicationVariable** from the list.

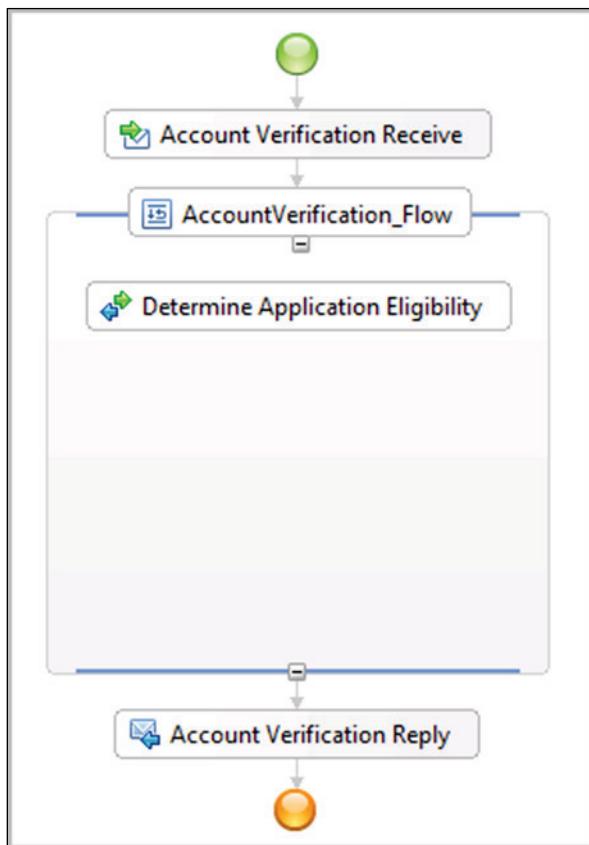
Note: Do not select CustomerApplicationVariable2.

8. In the **Outputs Store into Variable** column, click the **none** link and select **CustomerApplicationVariable** from the list.

	Name	Type	Read from Variable
Inputs	credappin	CustomerApplication	CustomerApplicationVariable
	Name	Type	Store into Variable
Outputs	credappout	CustomerApplication	CustomerApplicationVariable

9. Save your changes.

Notice that the errors in the Problems view go away. The process diagram resembles the one shown. (The sticky note was hidden in the following graphic by using the **Hide Notes** menu option.)



Note: You created the first activity Determine Application Eligibility from where the process flow starts. See the process model that is provided at the start of the lab. You implement this service invoke in the upcoming labs.

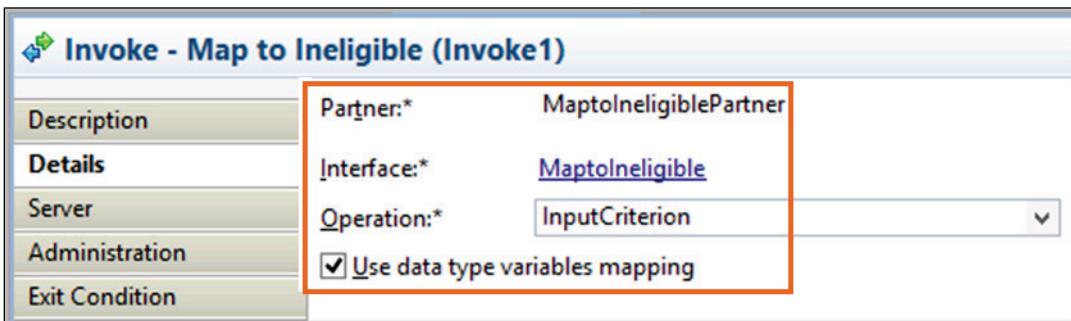
4. Add an invoke activity with the Display Name: Map to Ineligible

The Map to Ineligible invoke activity invokes the InputCriterion operation of the Maptoineligible interface of MaptoineligiblePartner.

If it is determined that the customer is not eligible for an account, the Map to Ineligible activity invokes a service to archive the application. In practice, this activity would likely be used to notify the customer as well.

1. In the palette, expand **Basic Actions** and select **Invoke**.
2. Click under **Determine Application Eligibility** to add the activity to the diagram.
3. In the **Properties** view on the **Description** tab, change the **Display Name** to: Map to Ineligible

4. Switch to the **Details** tab in the **Properties** view.
5. Beside the **Partner** field, click **Browse** and select **MaptoIneligiblePartner** in the “Select a Partner” dialog box.
6. Click **OK**.
7. Verify that the **Interface** field is set to **MaptoIneligible**, the **Operation** is set to **InputCriterion**, and the **Use data type variables mapping** check box is selected.



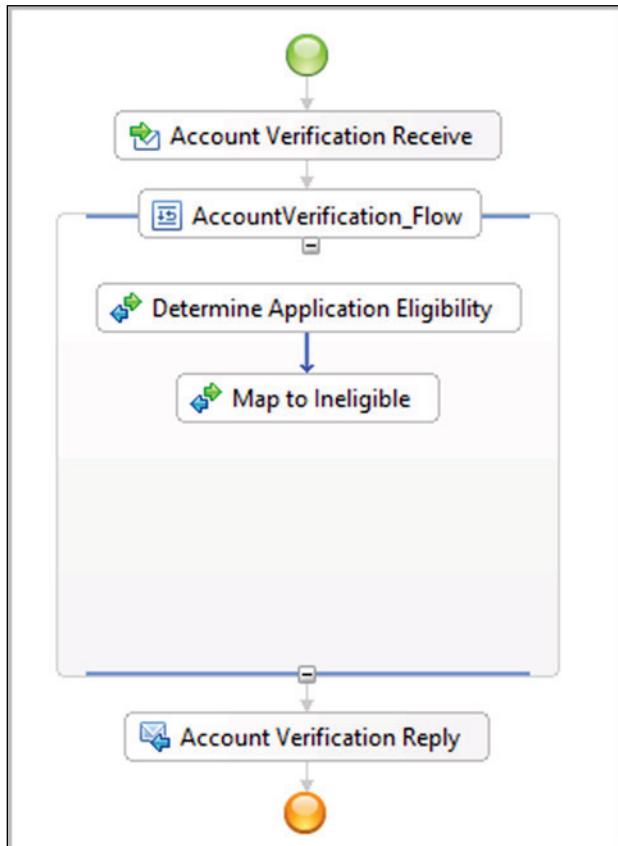
8. In the **Inputs Read from Variable** column, click the **none** link and select **CustomerApplicationVariable** from the list.
Note: Do **not** select **CustomerApplicationVariable2**.
9. In the **Outputs Store into Variable** column, click the **none** link and select **IneligibleApplicationVariable** from the list.

	Name	Type	Read from Variable	Store into Variable
Inputs	Input	CustomerApplication	CustomerApplicationVariable	IneligibleApplicationVariable
	Name	Type		
Outputs	Output	IneligibleApplication		

5. Link Determine Application Eligibility to Map to Ineligible.

1. Right-click Determine Application Eligibility and click Add a link from the menu.
2. Click **Map to Ineligible** to add the link.

Your diagram looks like the following figure:

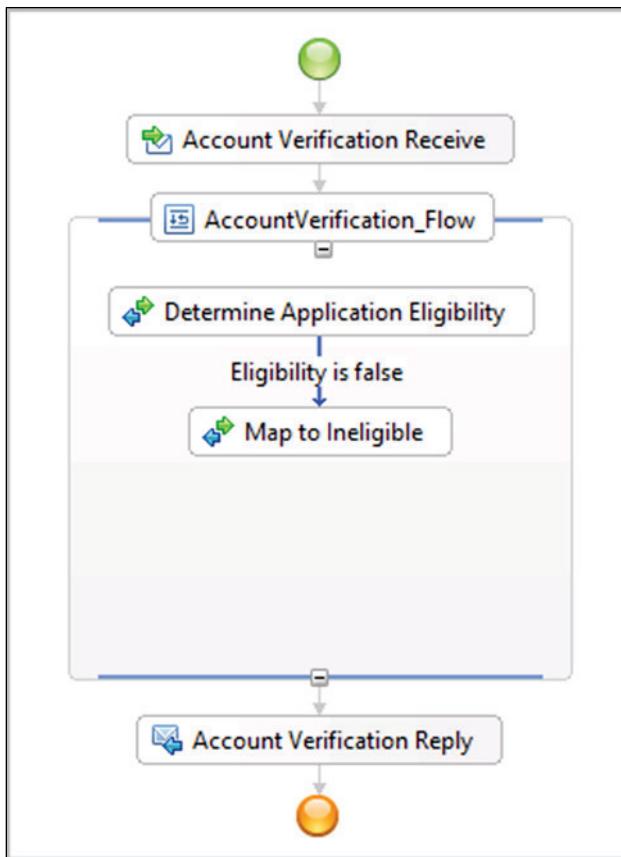


6. Change the Display Name of the link between Determine Application Eligibility and Map to Ineligible to: Eligibility is false

1. Select the link between the Determine Application Eligibility and Map to Ineligible.
2. Click the Description tab in the Properties view.
3. Change the Display Name of this link to: Eligibility is false

4. Right-click the process diagram and click **Show Labels on Links** from the menu.

The diagram looks like the following figure:

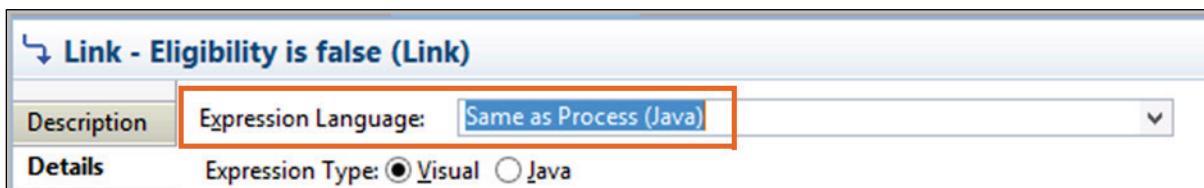


Troubleshooting: It is possible that the labels do not show on the links after saving your changes. If that occurs, then try turning off the label and then turning it back on again. If they still do not show, then you can continue with the lab because it has no impact.

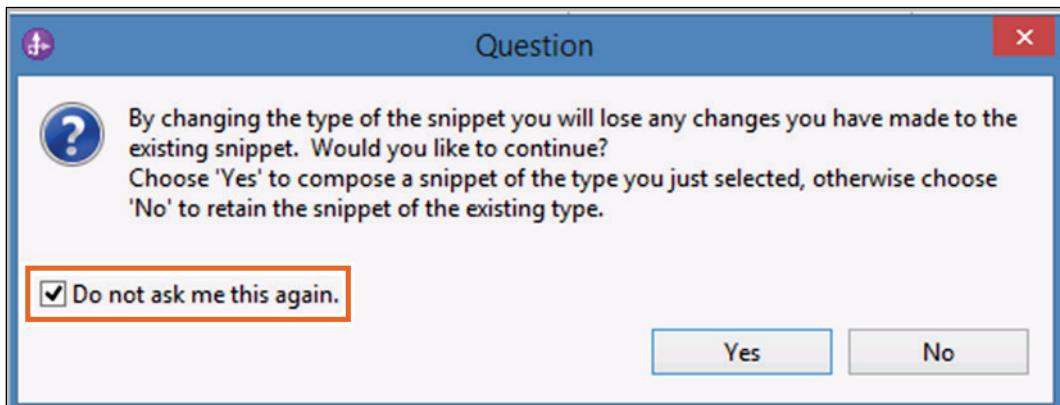
Note: Depending on when you save your process, you might see a red error indicator at the lower left of your process. Do not be concerned, even if the error is there during the entire exercise. However, the error should not be there at the end of this exercise.

7. Set the link condition so the process flows from Determine Application Eligibility to Map to Ineligible when the eligibleApplication attribute value is `false`.
1. Select the link between the Determine Application Eligibility and Map to Ineligible.

2. Switch to the **Details** tab in the **Properties** view.
3. In the **Expression Language** field, select **Same as Process (Java)**.



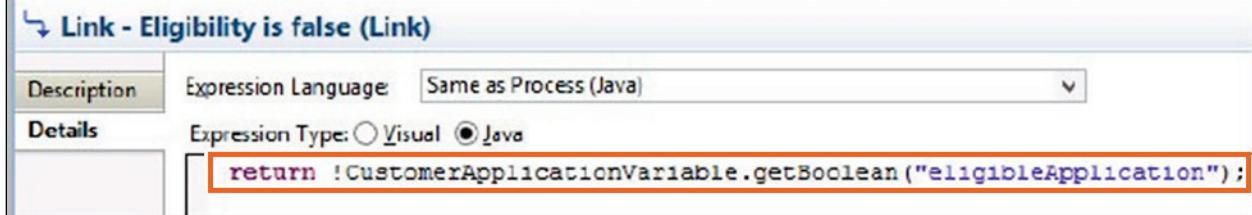
4. For **Expression Type**, click **Java**.
5. In the Question dialog box, select **Do not ask me this again** and click **Yes** to verify that you want to switch to a Java (text) expression.



6. Open Windows Explorer and browse to C:\labfiles\Support Files\Ex6.
7. Open AccountVerification_snippets.txt in a text editor such as Notepad.
This file contains all of the Java code that the different parts of your process use.
8. Copy the Determine App Eligibility --> Map to Ineligible code snippet. You do not need to copy the comment lines that begin with //.

```
//-
// Determine App Eligibility --> Map to Ineligible
//-
return !CustomerApplicationvariable.getBoolean("eligibleApplication");
```

9. Paste the snippet in the expression window over the existing text. Alternatively, you can manually replace the code with the following text:
return
!CustomerApplicationVariable.getBoolean("eligibleApplication");

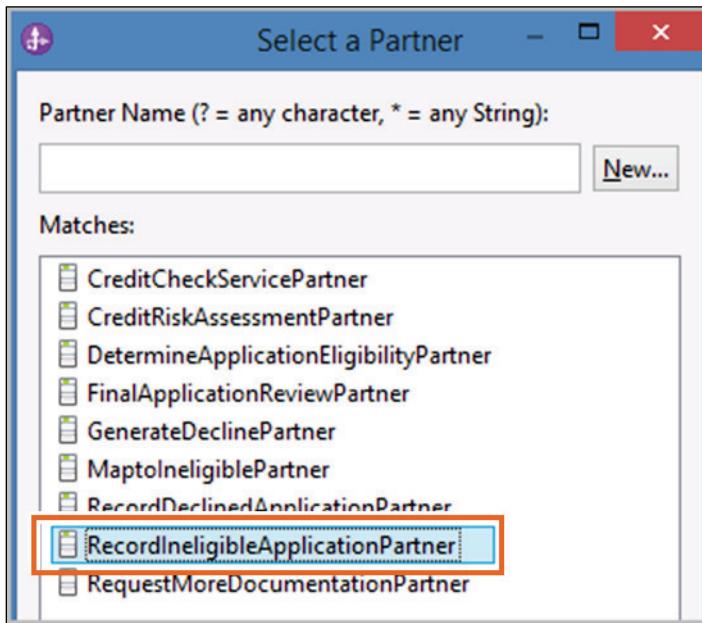


8. Add an invoke activity with the Display Name: Record Ineligible Application

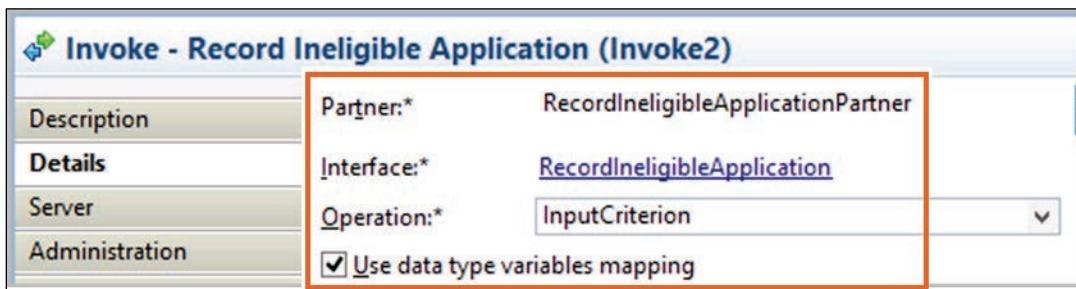
This activity invokes the InputCriterion operation of the RecordIneligible interface of RecordIneligibleApplicationPartner.

This activity archives any customer application that was determined to be ineligible. You implement the service that archives the application in a later exercise.

10. In the palette, expand **Basic Actions**, and select **Invoke**.
11. Click under **Map to Ineligible** to place the activity on the diagram.
12. On the **Description** tab in the **Properties** view, change the **Display Name** to: Record Ineligible Application
13. Switch to the **Details** tab in the **Properties** view.
14. Beside the **Partner** field, click **Browse** and select **RecordIneligibleApplicationPartner** in the “Select a Partner” dialog box.



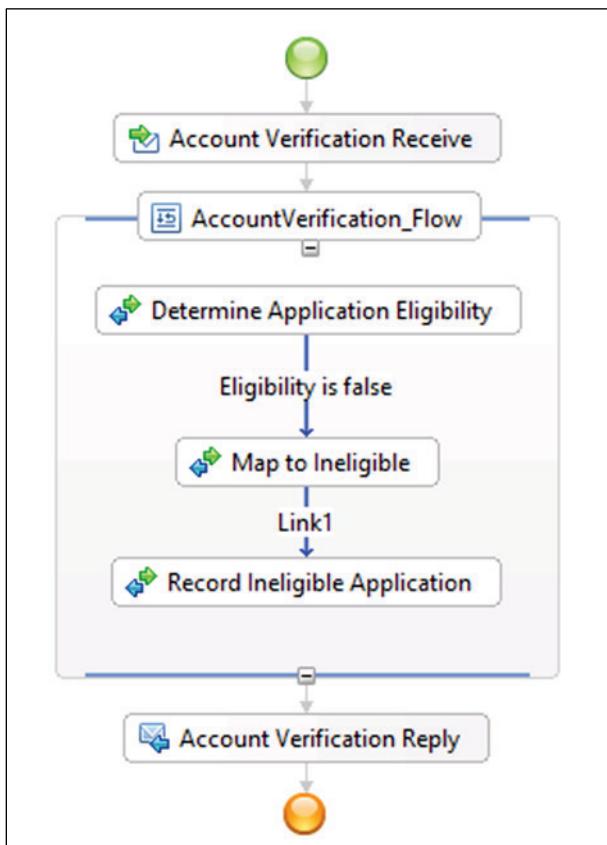
15. Click **OK**.
16. Verify that the **Interface** field is set to RecordIneligibleApplication, the **Operation** field is set to InputCriterion, and the **Use data type variables mapping** option is selected.



17. In the **Inputs Read from Variable** column, click the **none** link and select **IneligibleApplicationVariable** from the list.
18. In the **Outputs Store into Variable** column, click the **none** link and select **MessageVariable** from the list.

	Name	Type	Read from Variable
Inputs	Input	IneligibleApplication	IneligibleApplicationVariable
	Name	Type	Store into Variable
Outputs	Output	Message	MessageVariable

9. Link the Map to Ineligible activity to the Record Ineligible Application activity.
 1. Right-click **Map to Ineligible** and click **Add a link** from the menu.
 2. Click **Record Ineligible Application** to add the link.
 3. Accept the default link display name.
 4. To organize the contents for readability, right-click any blank space inside the **AccountVerification_Flow** activity and select **Align Parallel Activities Contents Automatically** (if not already checked).



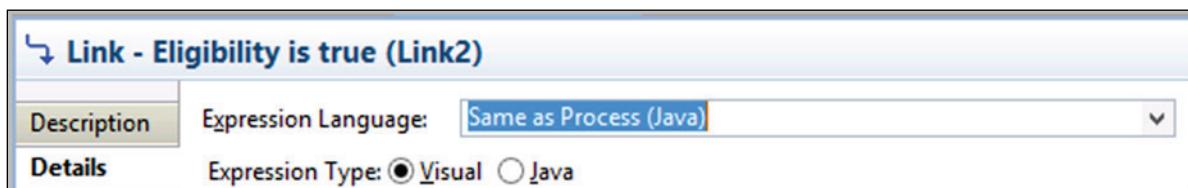
Underneath and to the right of **Determine Application Eligibility**, you add an empty action with the **Display Name**: Map to Credit Check

When implemented, Map to Credit Check transforms the data from a customer application into a format that is suitable for the credit score service. You are using an empty action because this activity is implemented in a later exercise.

5. In the palette, expand **Basic Actions** and select **Empty Action**.
6. Click under and to the right of **Determine Application Eligibility** to place the activity on the diagram.
7. In the **Properties** view on the **Description** tab, change the **Display Name** to: Map to Credit Check
10. Link Determine Application Eligibility to Map to Credit Check and change the Display Name of the link to: Eligibility is true
 1. Right-click Determine Application Eligibility and click Add a link from the menu.
 2. Click **Map to Credit Check** to add the link.
Accept the default link display name.
 3. Select the link between **Determine Application Eligibility** and **Map to Credit Check**.
 4. In the **Properties** view on the **Description** tab, change the **Display Name** to:

Eligibility is true

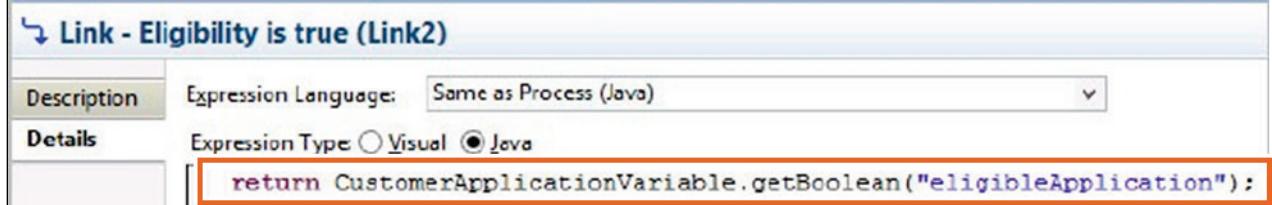
5. To organize the contents for readability, right-click any blank space inside the **AccountVerification_Flow** activity and select **Align Parallel Activities Contents Automatically** (if not already checked).
11. Set the link condition so that the process flows from Determine Application Eligibility to Map to Credit Check when the eligibleApplication attribute value is true.
 1. With the link between Determine Application Eligibility and Map to Credit Check selected, switch to the Details tab in the Properties view.
 2. In the **Expression Language** field, select **Same as Process (Java)**.



3. For **Expression Type**, click **Java**.
4. Copy the Determine App Eligibility --> Map to Credit Check code snippet from C:\labfiles\Support Files\Ex6\AccountVerification_snippets.txt. You do not need to copy the comment lines that begin with //.

```
// Determine App Eligibility --> Map to Credit Check
-----
return CustomerApplicationvariable.getBoolean("eligibleApplication");
```

5. Paste the snippet in the expression window over the existing text. Alternatively, you can manually replace the code with the following text:
return
CustomerApplicationVariable.getBoolean("eligibleApplication");



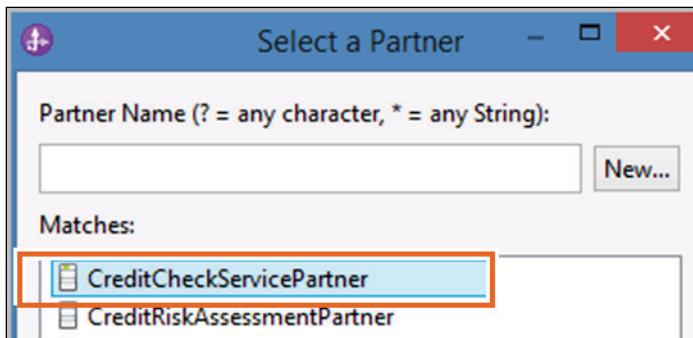
Be sure to leave the snippets file open.

12. Add an invoke activity under Map to Credit Check with the Display Name: Credit Check Service

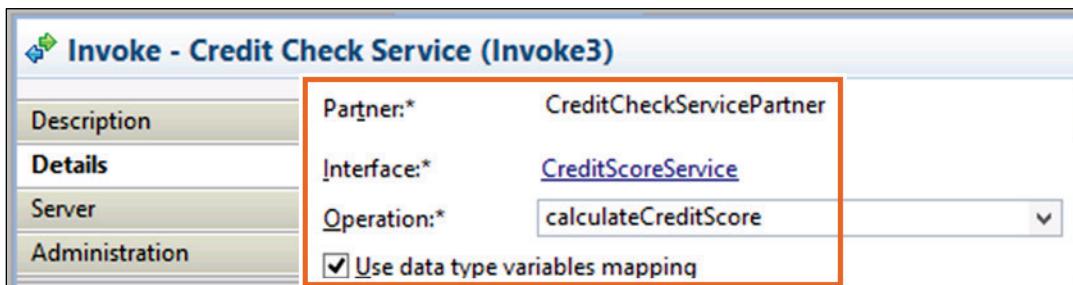
The Credit Check Service activity invokes the calculateCreditScore operation of the CreditScoreService interface of CreditCheckServicePartner.

When the application data is transformed into a format that is suitable for the credit score service, the application is sent to the service for credit score calculation. In this application, business rules determine the credit scores. In practice, this service is likely to be an external service that a Business Partner offers.

6. In the palette, expand **Basic Actions** and select **Invoke**.
7. Click under **Map to Credit Check** to place the activity on the diagram.
8. In the **Properties** view on the **Description** tab, change the **Display Name** to: Credit Check Service
9. Switch to the **Details** tab in the **Properties** view.
10. Beside the **Partner** field, click **Browse** and select **CreditCheckServicePartner** from the “Select a Partner” dialog box.



11. Click **OK**.
12. Verify that **Interface** is set to CreditScoreService, **Operation** is set to calculateCreditScore, and the **Use data type variables mapping** check box is selected.



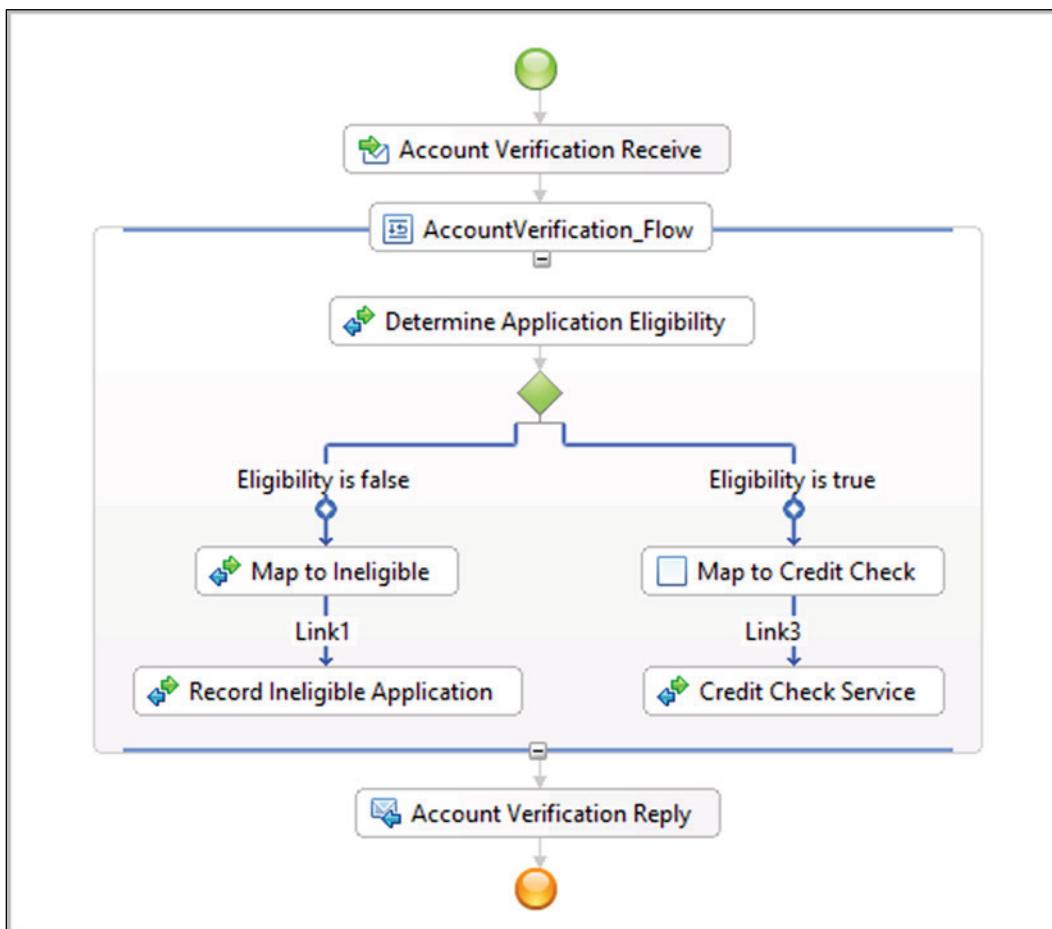
13. In the **Inputs Read from Variable** column, click the **none** link and select **CreditCheckVariable** from the list.
14. In the **Outputs Store into Variable** column, click the **none** link and select **CreditCheckVariable** from the list.

	Name	Type	
Inputs	request	CreditCheckRequest	Read from Variable CreditCheckVariable 
	Name	Type	Store into Variable  CreditCheckVariable
Outputs	calculateCreditScoreReturn	CreditCheckRequest	 CreditCheckVariable

13. Link Map to Credit Check to Credit Check Service.

1. Right-click **Map to Credit Check** and click **Add a link** from the menu.
2. Click **Credit Check Service** to add the link.

Accept the default link display name. The diagram looks similar to the following figure:



Note: See the model at the beginning of the lab. The **Perform Credit Check** activity in the model corresponds to the **Credit Check Service** build. The invoked service is implemented in the upcoming labs.

14. Add an empty action under Credit Check Service with the Display Name: Map Credit Checking Result

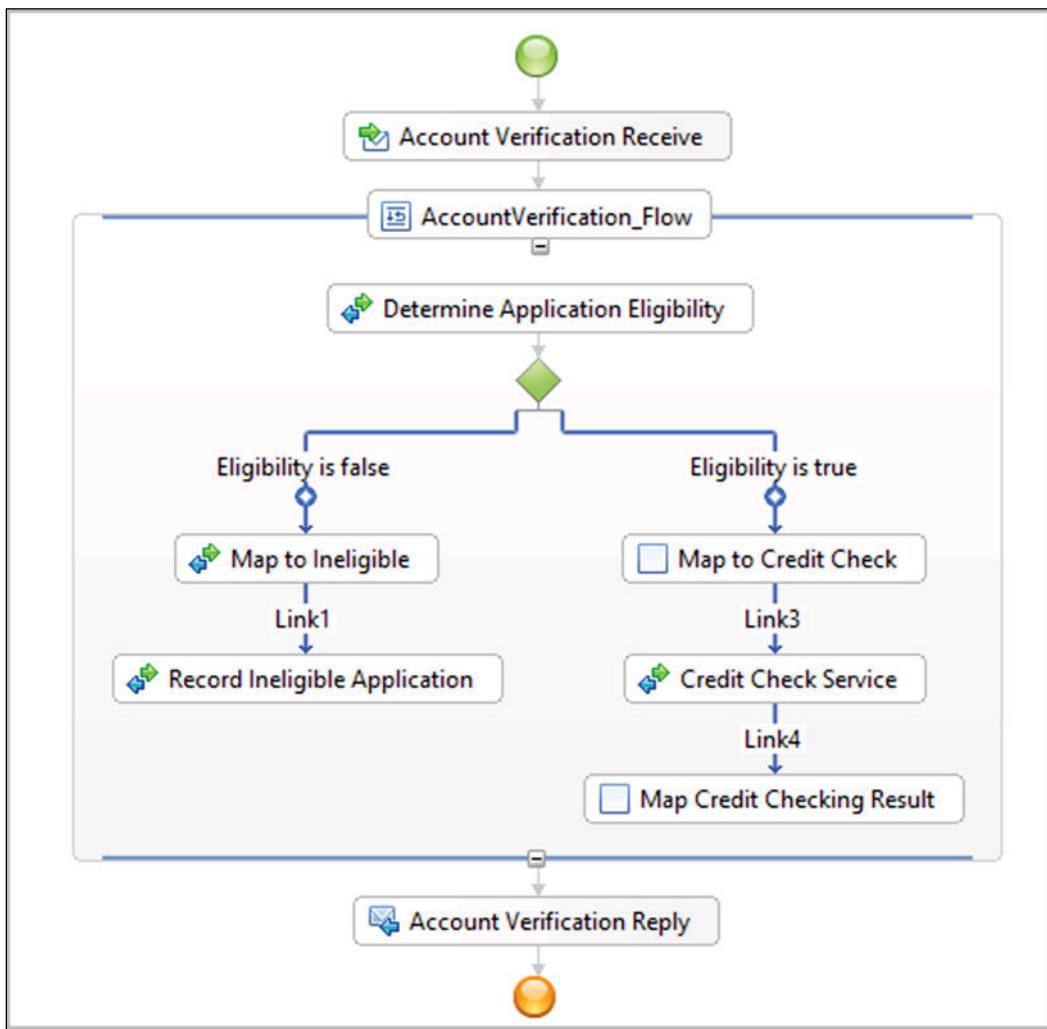
When the credit score is returned, the data is again transformed. In this case, it is transformed back into a customer application. You are using an empty action because this activity is implemented in a later exercise.

1. In the palette, expand **Basic Actions** and select **Empty Action**.
2. Click under **Credit Check Service** to place the activity on the diagram.
3. In the **Properties** view on the **Description** tab, change the **Display Name** to: Map Credit Checking Result

15. Link Credit Check Service to Map Credit Checking Result.

1. Right-click **Credit Check Service** and click **Add a link** from the menu.
2. Click **Map Credit Checking Result** to add the link.

Accept the default link display name. The diagram looks like the following figure:



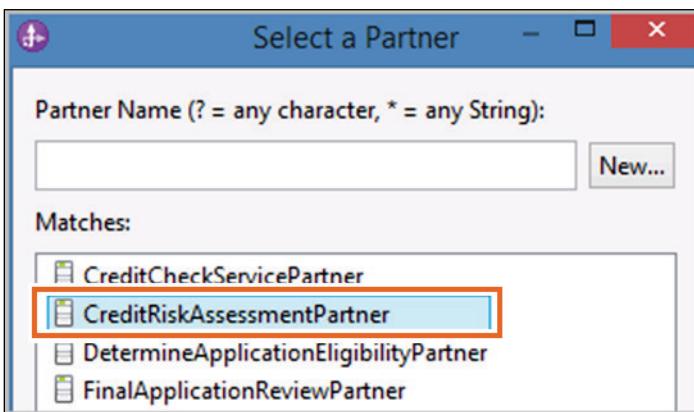
16. Add an invoke activity under Map Credit Checking Result with Display Name: Credit Risk Assessment

Credit Risk Assessment invokes the InputCriterion operation of the CreditRiskAssessment interface of CreditRiskAssessmentPartner. The CreditRiskAssessment activity examines the credit score that is received and takes one of three actions.

If the credit risk is high, more documentation is requested before final review. If the credit risk is medium, final employee review is requested before approval. If the credit risk is low, the application is approved.

1. In the palette, expand **Basic Actions** and select **Invoke**.
2. Click under **Map Credit Checking Result** to place the activity on the diagram.
3. In the **Properties** view on the **Description** tab, change the **Display Name** to: Credit Risk Assessment
4. Switch to the **Details** tab in the **Properties** view.

5. Beside the **Partner** field, click **Browse** and select **CreditRiskAssessmentPartner** from the “Select a Partner” dialog box.



6. Click **OK**.
7. Verify that the **Interface** field is set to **CreditRiskAssessment**, the **Operation** is set to **InputCriterion**, and the **Use data type variables mapping** check box is selected.

Partner: [*]	CreditRiskAssessmentPartner
Interface: [*]	CreditRiskAssessment
Operation: [*]	InputCriterion
<input checked="" type="checkbox"/> Use data type variables mapping	

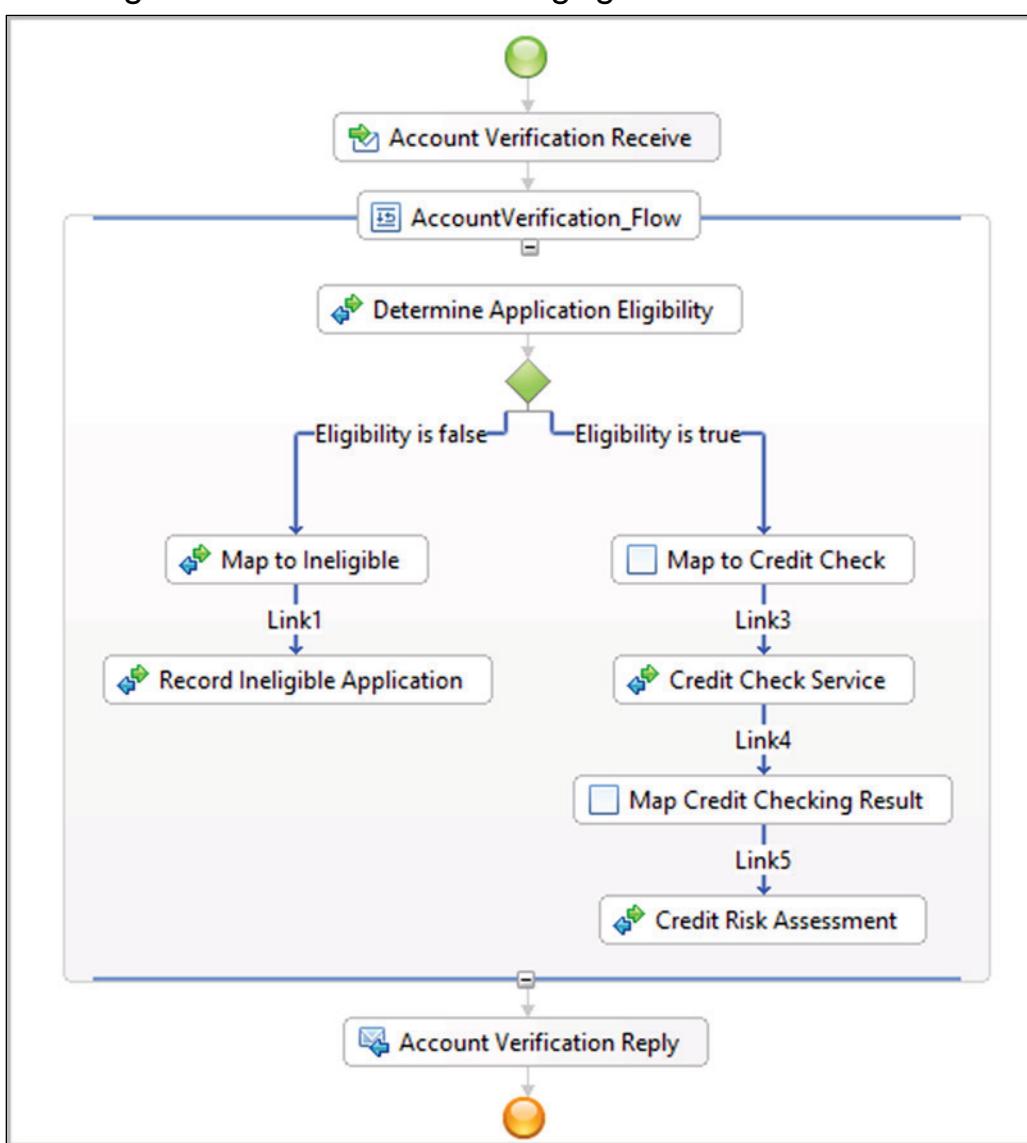
8. In the **Inputs Read from Variable** column, click the **none** link and select **CustomerApplicationVariable** from the list.
Note: Do not select CustomerApplicationVariable2.
9. In the **Outputs Store into Variable** column, click the **none** link and select **CustomerApplicationVariable** from the list.

	Name	Type	Read from Variable	Store into Variable
Inputs	Input	CustomerApplication	CustomerApplicationVariable	CustomerApplicationVariable
	Name	Type		
Outputs	Output	CustomerApplication	CustomerApplicationVariable	CustomerApplicationVariable

17. Link Map Credit Checking Result to Credit Risk Assessment.

1. Right-click Map Credit Checking Result and click Add a link from the menu.

2. Click **Credit Risk Assessment** to add the link.
Accept the default link display name.
3. To organize the contents for readability, right-click any blank space inside the **AccountVerification_Flow** activity and click **Align Parallel Activities Contents Automatically** (if not already selected).
The diagram looks like the following figure:



4. Save your changes.
18. Add a snippet activity under Credit Risk Assessment with the Display Name: Assign Variable
Assign Variable copies CustomerApplicationVariable to CustomerApplicationVariable2. Before writing the customer application business object into the archive, this code copies the business object into a new variable.

 1. In the palette, expand **Basic Actions** and select **Snippet**.

2. Click under **Credit Risk Assessment** to add the snippet to the diagram.
3. In the **Properties** view on the **Description** tab, change the **Display Name** to: Assign Variable
4. Switch to the **Details** tab in the **Properties** view.
5. Click **Java**.
6. Copy the "Assign Variable" snippet code from C:\labfiles\Support Files\Ex6\ AccountVerification_snippets.txt. The file is still open.

```
// "Assign variable" snippet
//*****snip*****
if (CustomerApplicationVariable2 == null){
    com.ibm.websphere.bo.BOFactory factory =
(com.ibm.websphere.bo.BOFactory) new com.ibm.websphere.sca.ServiceManager()
.locateService("com/ibm/websphere/bo/BOFactory");
    CustomerApplicationVariable2 = factory.create(
"http://FoundationLibrary/creditserviceitems", "CustomerApplication");
}
CustomerApplicationVariable2=CustomerApplicationvariable;
```

Note: This snippet can be replaced with an Assign activity. The purpose of using it as written is to show you the BOFactory and ServiceManager().locateService() APIs.

7. Paste the snippet in the expression window. Alternatively, you can enter the following text:

```
if (CustomerApplicationVariable2 == null){
com.ibm.websphere.bo.BOFactory factory =
(com.ibm.websphere.bo.BOFactory) new
com.ibm.websphere.sca.ServiceManager().locateService("com/ibm/websph
ere/b o/BOFactory");
CustomerApplicationVariable2 = factory.create(
"http://FoundationLibrary/creditserviceitems",
"CustomerApplication");
}
CustomerApplicationVariable2=CustomerApplicationVariable;
```

8. Save your changes. Continue to ignore any errors in the **Problems** view.
19. Link Credit Risk Assessment to Assign Variable and change the link Display Name to:
Credit risk is HIGH

 1. Right-click **Credit Risk Assessment** and click **Add a link** from the menu.
 2. Click **Assign Variable** to add the link.
 3. In the **Properties** view on the **Description** tab, change the **Display Name** to:
Credit risk is HIGH

20. Set the link condition so the process flows from Credit Risk Assessment to Assign Variable when the customer's creditRisk is HIGH.

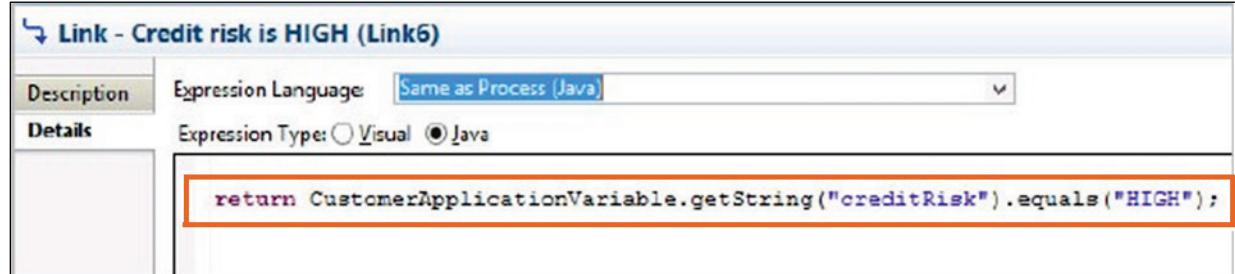
If the credit risk is HIGH, more steps are taken to gather more documentation before reaching Final Application Review.

1. Select the link between Credit Risk Assessment and Assign Variable.
2. Switch to the **Details** tab in the **Properties** view.
3. In the **Expression Language** field, select **Same as Process (Java)**.
4. For **Expression Type**, click **Java**.
5. Copy the Credit Risk Assessment --> Assign Variable code snippet from C:\labfiles\Support Files\Ex6\AccountVerification_snippets.txt.

```
// Credit Risk Assessment --> Assign Variable
//-----
return CustomerApplicationVariable.getString("creditRisk").equals("HIGH");
```

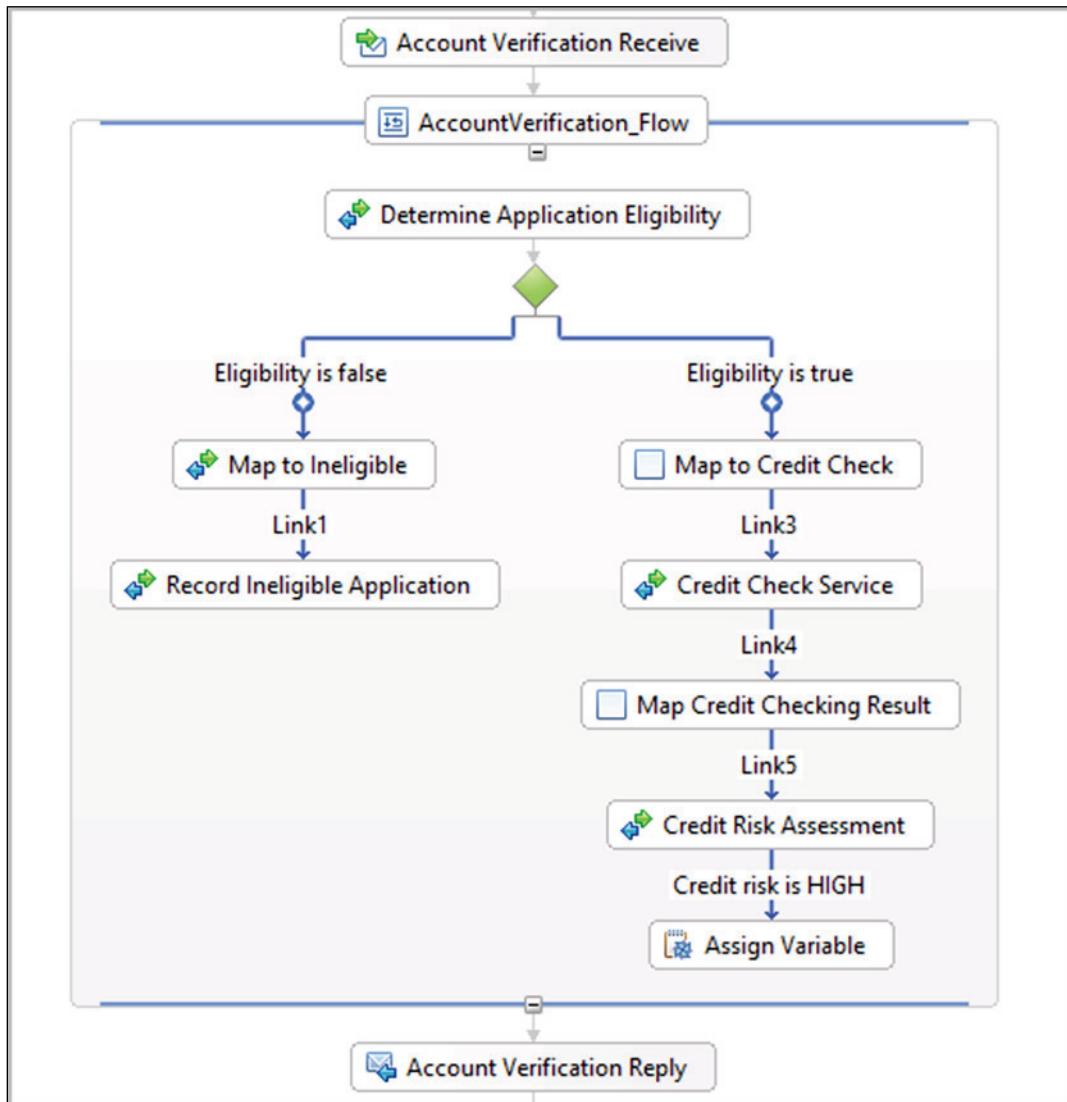
6. Paste the snippet in the expression window over the existing text. Alternatively, you can replace the code with the following text:

```
return
CustomerApplicationVariable.getString("creditRisk").equals("HIGH");
```



7. Save your changes. To organize the contents for readability, right-click any blank space inside the **AccountVerification_Flow** activity and click **Align Parallel Activities Contents Automatically**.

The diagram looks like the following figure:



21. Add a While Loop activity under Assign Variable with the Display Name: While More Documents Required

The While More Documents Required loop continues to request documentation while the comment attribute is equal to None. When an employee reviews the application, the comment attribute is populated and the loop ends.

1. Expand **Structures** in the palette and select **While Loop**.
2. Click under **Assign Variable** to add the activity to the diagram.
3. In the **Properties** view on the **Description** tab, change the **Display Name** of the loop to: While More Documents Required
4. Switch to the **Details** tab in the **Properties** view.
5. In the **Expression Language** field, select **Same as Process (Java)**.

6. For **Expression Type**, click **Java**.
7. Copy the "While Loop" condition code from **C:\labfiles\Support Files\Ex6\ AccountVerification_snippets.txt**.

```
/*
 * "while Loop" condition
 */
return CustomerApplicationvariable2.getString("comments").equals("None");
```

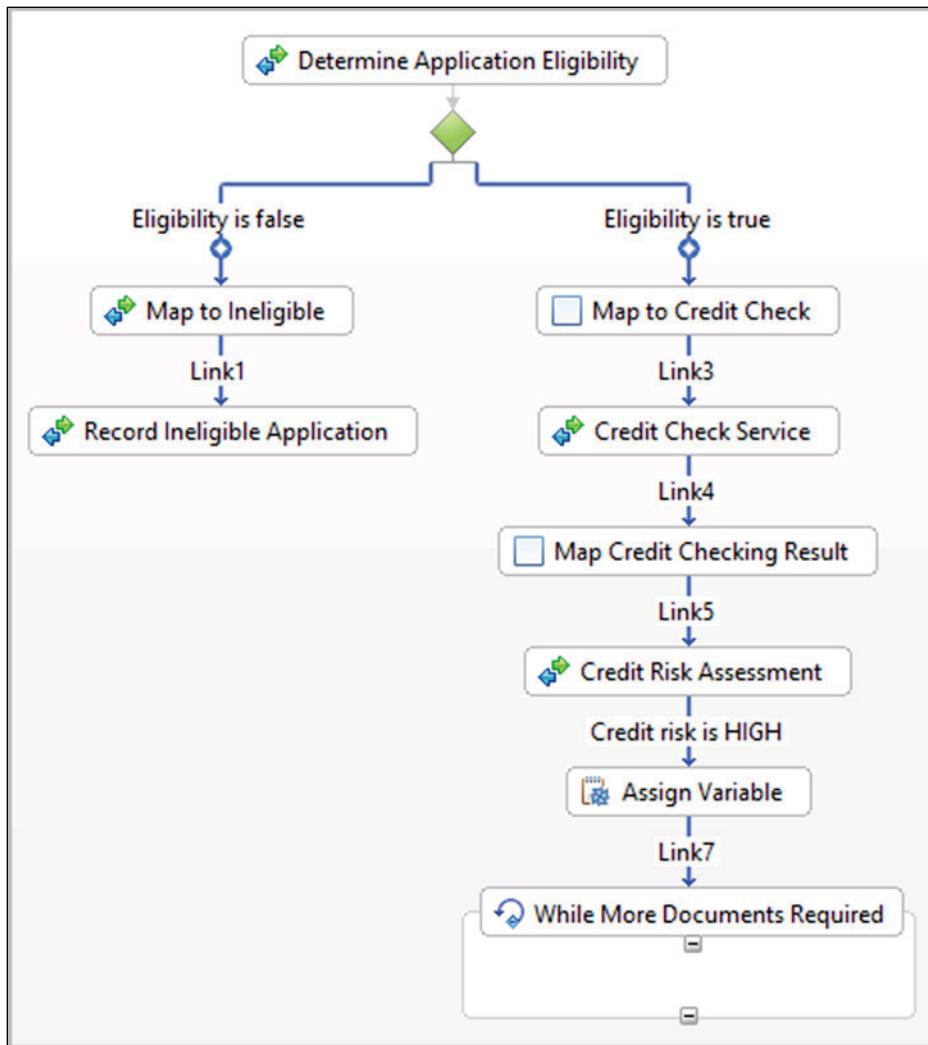
8. Paste the snippet in the expression window over the existing text. Alternatively, you can replace the code with the following text:

```
return
CustomerApplicationVariable2.getString( "comments" ) .equals( "None" );
```

22. Link Assign Variable to While More Documents Required.

1. Right-click **Assign Variable** and click **Add a link** from the menu.
2. Click **While More Documents Required** to add the link.
3. Accept the default link display name.

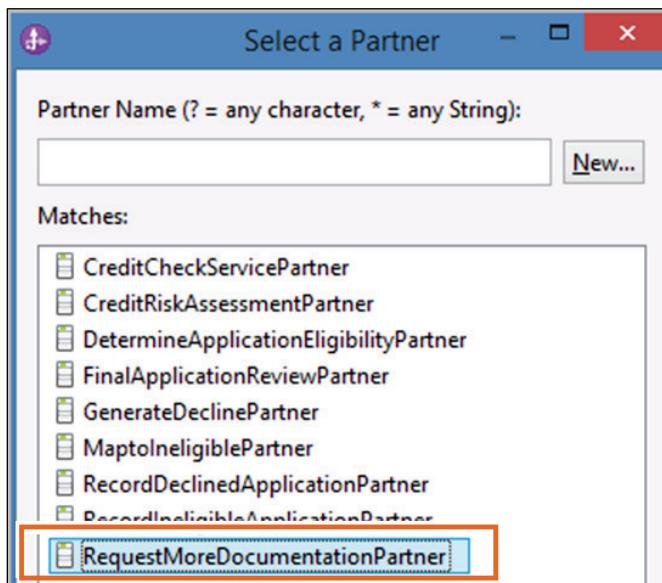
The current path in your diagram looks like the following figure:



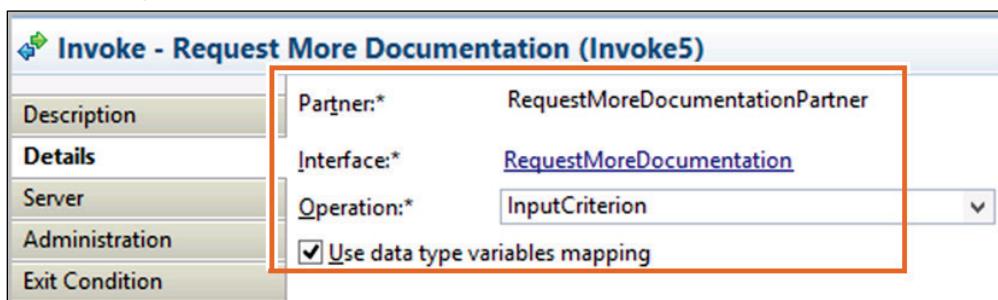
23. Add an invoke activity inside the while loop with the Display Name: Request More Documentation

The Request More Documentation activity invokes the InputCriterion operation of the RequestMoreDocumentation interface of RequestMoreDocumentationPartner. You implement the human task that is associated with this activity in a later exercise.

1. In the palette, expand **Basic Actions** and select **Invoke**.
2. Click inside the while loop to add the activity to the diagram.
3. In the **Properties** view on the **Description** tab, change the **Display Name** to: Request More Documentation
4. Switch to the **Details** tab in the **Properties** view.
5. Beside the **Partner** field, click **Browse** and select **RequestMoreDocumentationPartner** from the “Select a Partner” dialog box.



6. Click **OK**.
7. Verify that the **Interface** field is set to RequestMoreDocumentation, the **Operation** field is set to InputCriterion, and the **Use data type variables mapping** option is selected.

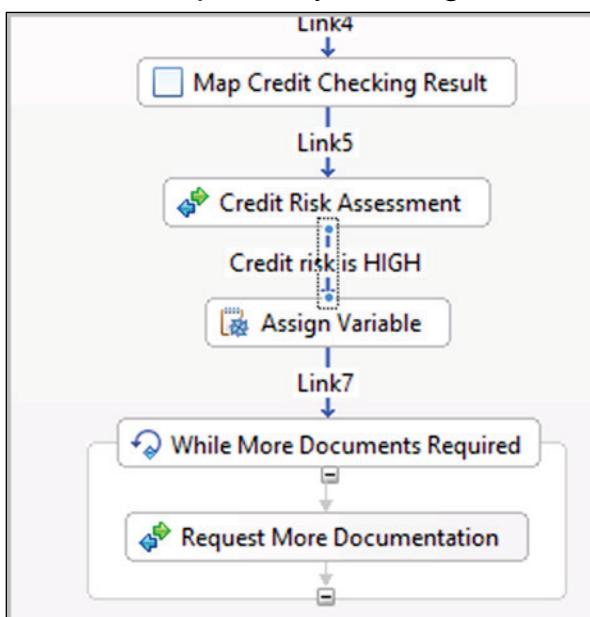


8. In the **Inputs Read from Variable** column, click the **none** link and select **CustomerApplicationVariable2** from the list.
9. In the **Outputs Store into Variable** column, click the **none** link and select **CustomerApplicationVariable2** from the list.

	Name	Type	Read from Variable
Inputs	Input	CustomerApplication	CustomerApplicationVariable2 
	Name	Type	Store into Variable
Outputs	Output	CustomerApplication	 CustomerApplicationVariable2

10. Save your changes.

The current path in your diagram looks like the following figure:



Note: See the model at the beginning of the lab. The More Documentation Required activity in the model corresponds to the preceding Request More Documentation invoke. The invoked service is implemented in the upcoming labs.

24. Add a snippet activity after the while loop with the Display Name: Merge Assign

The Merge Assign activity merges the data from the Request More Documentation service. This activity can also be implemented as an assign.

1. In the palette, expand **Basic Actions** and select **Snippet**.
2. Click beneath the **While More Documents Required** loop to add the snippet to the diagram.
3. In the **Properties** view on the **Description** tab, change the **Display Name** to: Merge Assign

4. Switch to the **Details** tab in the **Properties** view.
5. Click **Java**.
6. Copy the "Merge Assign" snippet code from C:\labfiles\Support Files\Ex6\ AccountVerification_snippets.txt.

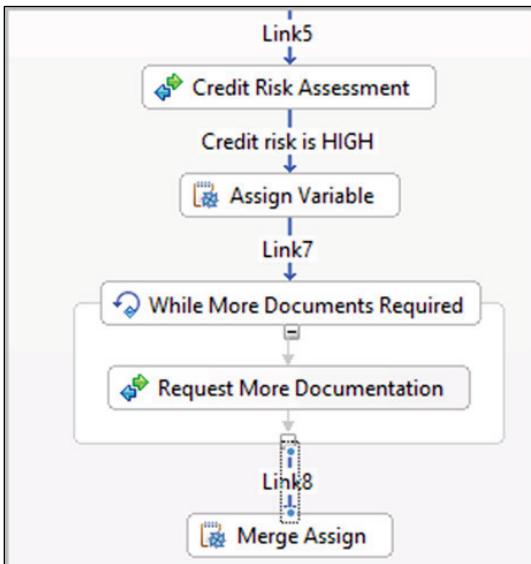
```
/*
 * "Merge Assign" snippet
 */
if (CustomerApplicationvariable == null){
    com.ibm.websphere.bo.BOFactory factory = (com.ibm.websphere.bo.BOFactory) new
    com.ibm.websphere.sca.ServiceManager().locateService("com/ibm/websphere/bo/BOFactory");
    CustomerApplicationVariable = factory.create(
        "http://FoundationLibrary/creditserviceitems", "CustomerApplication");
}
CustomerApplicationvariable=CustomerApplicationvariable2;
```

7. Paste the snippet in the expression window. Alternatively, you can replace the code with the following text:

```
if (CustomerApplicationVariable == null){
    com.ibm.websphere.bo.BOFactory factory =
        (com.ibm.websphere.bo.BOFactory) new
        com.ibm.websphere.sca.ServiceManager().locateService("com/ibm/websphere/b o/BOFactory");
    CustomerApplicationVariable = factory.create(
        "http://FoundationLibrary/creditserviceitems",
        "CustomerApplication" );
}
CustomerApplicationVariable=CustomerApplicationVariable2;
```

25. Link While More Documents Required to Merge Assign.

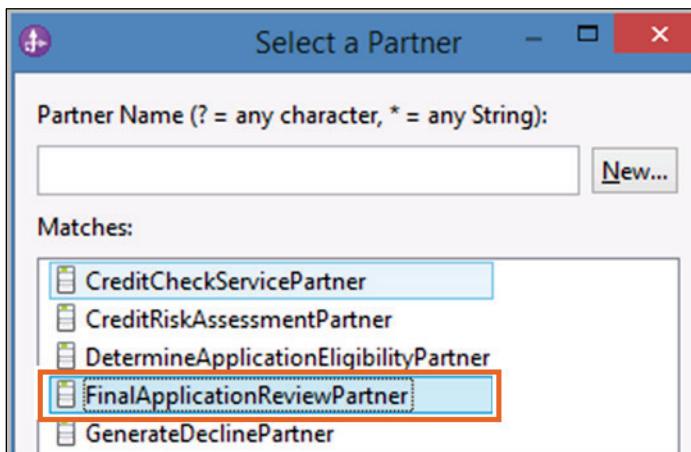
1. Right-click While More Documents Required and click Add a link from the menu.
2. Click **Merge Assign** to add the link.
3. Accept the default link display name, and Save your changes.



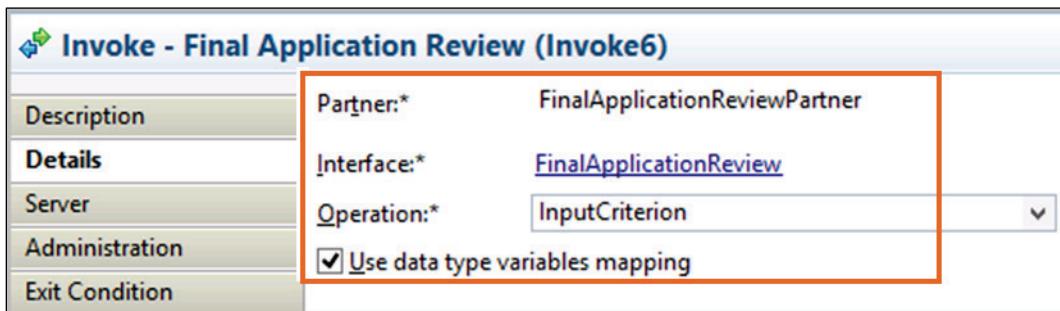
26. Add an invoke activity under Merge Assign with the Display Name: Final Application Review

The Final Application Review activity invokes the InputCriterion operation of the FinalApplicationReview interface of FinalApplicationReviewPartner. This activity invokes a human task that allows an employee to review an application. You implement the human task in a later exercise.

1. In the palette, expand **Basic Actions** and select **Invoke**.
2. Click under **Merge Assign** to add the activity to the diagram.
3. In the **Properties** view on the **Description** tab, change the **Display Name** to: Final Application Review
4. Switch to the **Details** tab in the **Properties** view.
5. Beside the **Partner** field, click **Browse** and select **FinalApplicationReviewPartner** from the “Select a Partner” dialog box.



6. Click **OK**.
7. Verify that the **Interface** field is set to **FinalApplicationReview**, the **Operation** field is set to **InputCriterion**, and the **Use data type variables mapping** check box is selected.



8. In the **Inputs Read from Variable** column, click the **none** link and select **CustomerApplicationVariable** from the list.

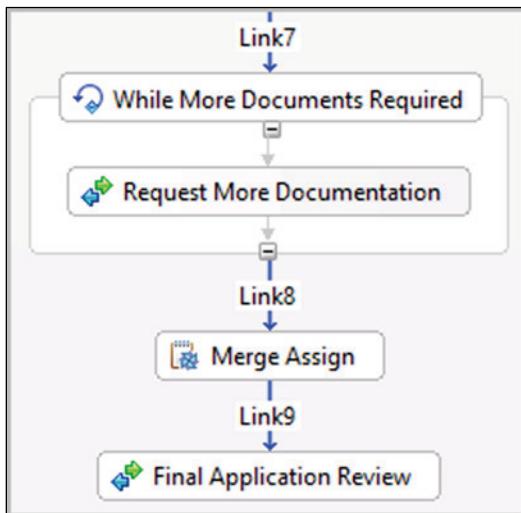
9. In the **Outputs Store into Variable** column, click the **none** link and select **CustomerApplicationVariable** from the list.

	Name	Type	
Inputs	Input	CustomerApplication	Read from Variable
	Name	Type	CustomerApplicationVariable
Outputs	Output	CustomerApplication	Store into Variable CustomerApplicationVariable

10. Save your changes. Continue to ignore any errors in the **Problems** view.
 27. Link Merge Assign to Final Application Review.

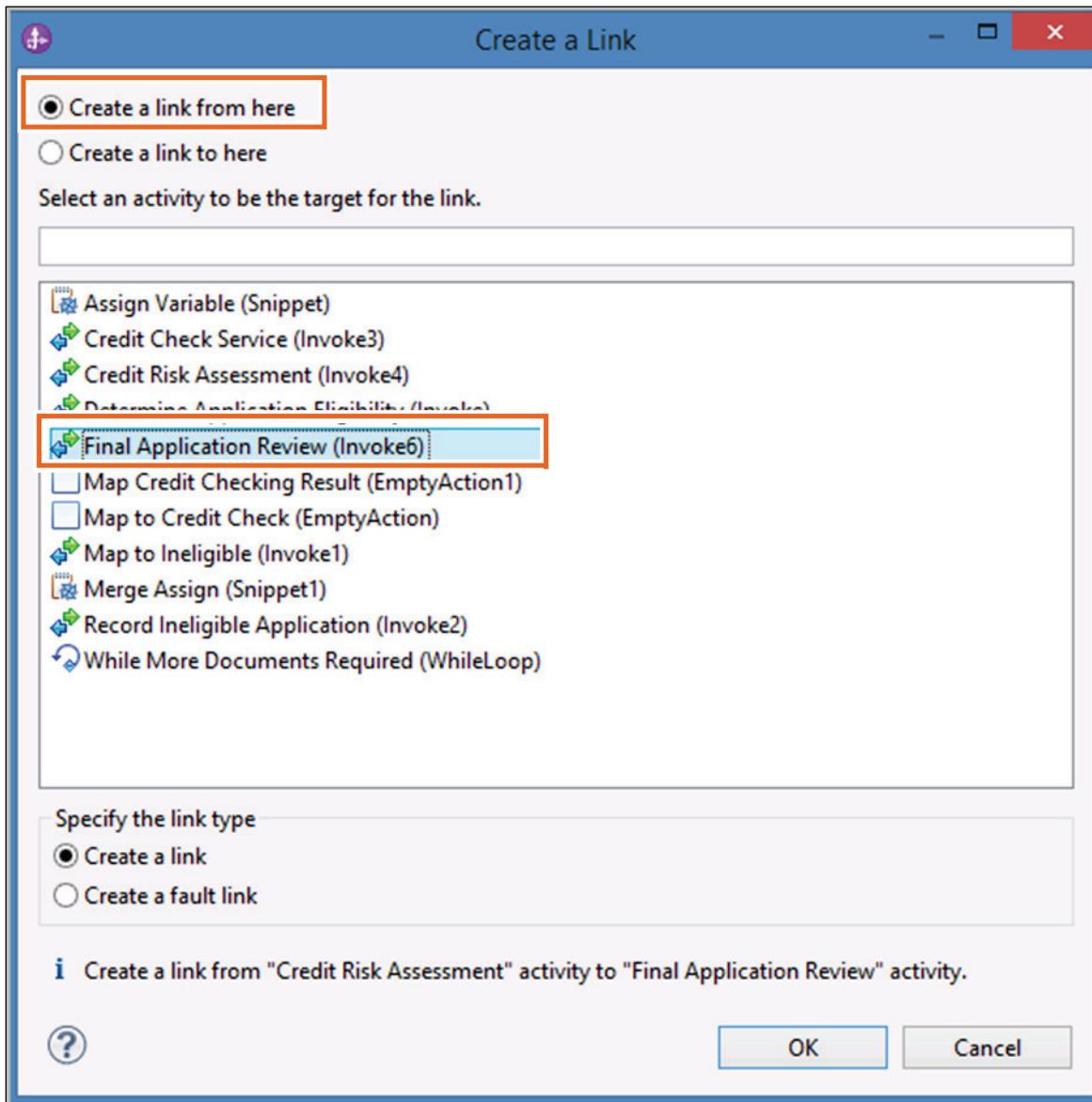
1. Right-click **Merge Assign** and click **Add a link** from the menu.
2. Click **Final Application Review** to add the link.
3. Accept the default link display name and save your changes.

The current path in your process looks like the following diagram:



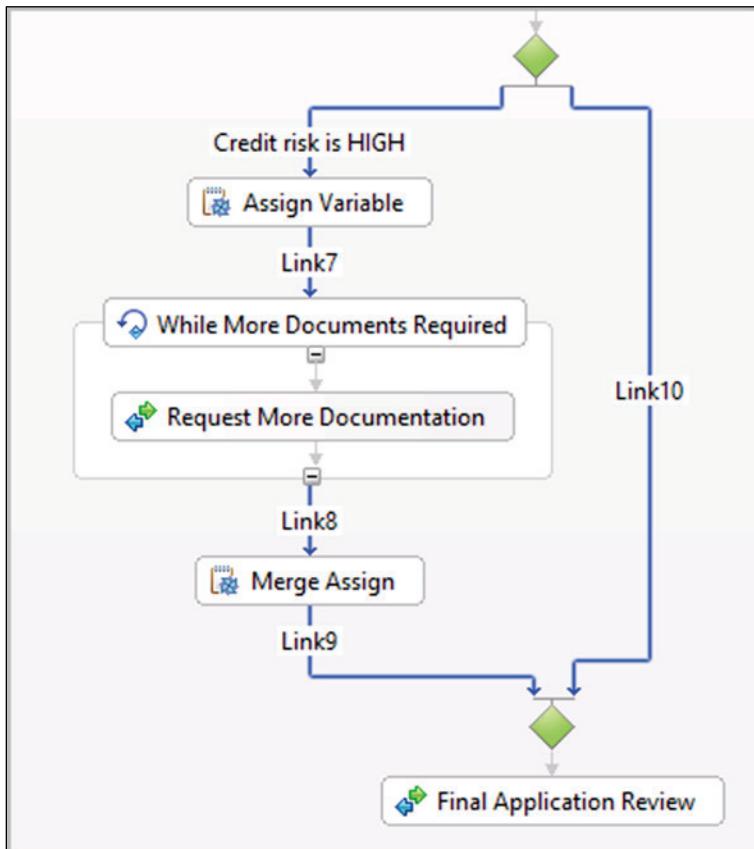
28. Link Credit Risk Assessment to Final Application Review.
1. Right-click Credit Risk Assessment and click **Create a Link From/To Here** from the menu.
 This option can be used to link to activities that are outside your viewing area.

2. In the “Create a Link” dialog box, take the following actions:
- Verify that **Create a link from here** is selected.
 - Select **Final Application Review** from the list.
 - Verify that **Create a link** is selected in the **Specify the link type** section.



3. Click **OK**, and accept the default link display name.

The current path in your process looks like the following diagram:



29. Set the link condition between Credit Risk Assessment and Final Application Review so that the process flows directly to Final Application Review when creditRisk is MED (short for medium). Change the link Display Name to: Credit risk is MED

If the credit risk is MED, the application is sent directly to Final Application Review. No additional documentation is required, but the credit risk is not low enough to warrant preapproval of the application without review.

1. Click the link between Credit Risk Assessment and Final Application Review.
2. In the **Properties** view on the **Description** tab, change the **Display Name** to: Credit risk is MED
3. Switch to the **Details** tab in the **Properties** view.
4. In the **Expression Language** field, select **Same as Process (Java)**.
5. For **Expression Type**, click **Java**.

6. Copy the Credit Risk Assessment --> Final Application Review code snippet from C:\labfiles\Support Files\Ex6\AccountVerification_snippets.txt.

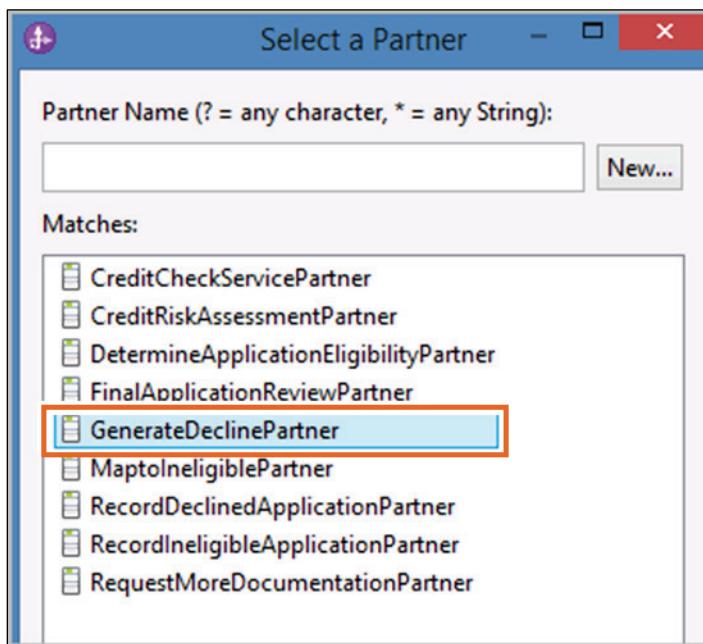
```
// Credit Risk Assessment --> Final Application Review
-----
return CustomerApplicationvariable.getString("creditRisk").equals("MED");
```

7. Paste the snippet in the expression window over the existing text. Alternatively, you can manually replace the code with the following text:
- ```
return
CustomerApplicationVariable.getString("creditRisk").equals("MED");
```
8. Save your changes.

30. You will add an invoke activity under Final Application Review with the Display Name: Generate Decline

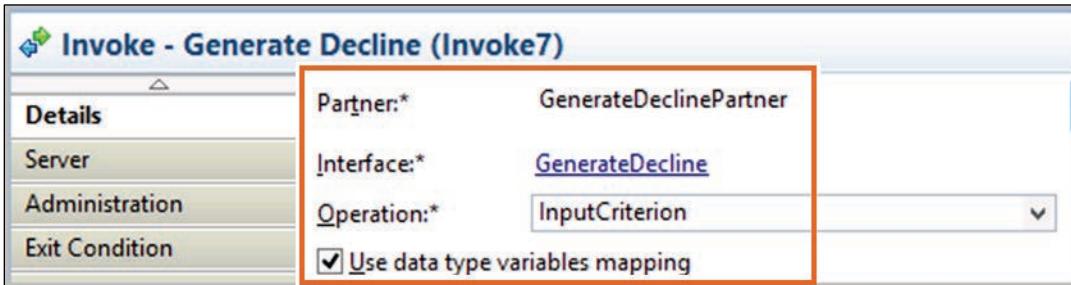
The Generate Decline activity invokes the InputCriterion operation of the GenerateDecline interface of GenerateDeclinePartner.

1. In the palette, expand **Basic Actions** and select **Invoke**.
2. Click under **Final Application Review** to add the activity to the diagram.
3. In the **Properties** view on the **Description** tab, change the **Display Name** to: Generate Decline
4. Switch to the **Details** tab in the **Properties** view.
5. Beside the **Partner** field, click **Browse** and select **GenerateDeclinePartner** from the “Select a Partner” dialog box.



6. Click **OK**.

7. Verify that the **Interface** field is set to `GenerateDecline`, the **Operation** is set to **InputCriterion**, and the **Use data type variables mapping** option is selected.



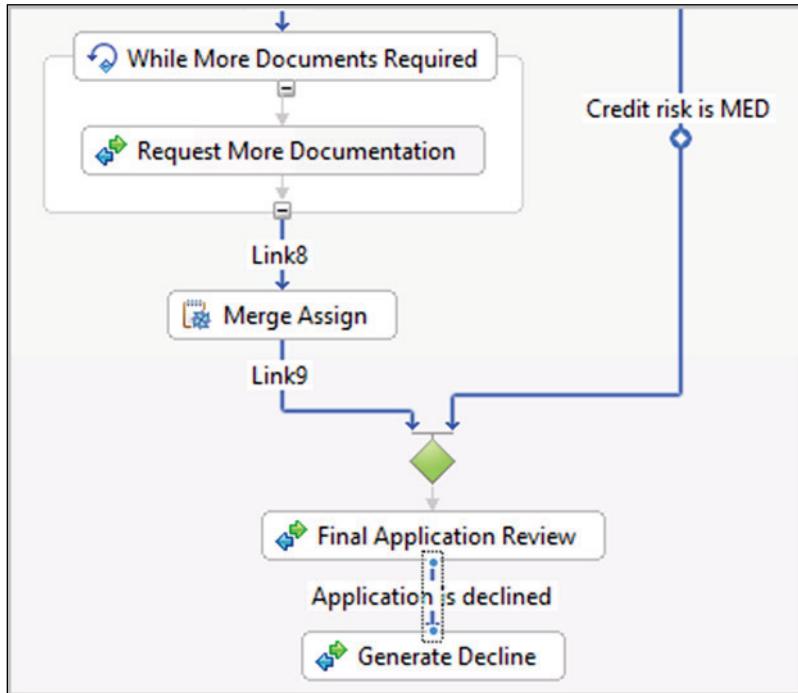
8. In the **Inputs Read from Variable** column, click the **none** link and select **CustomerApplicationVariable** from the list.  
 9. In the **Outputs Store into Variable** column, click the **none** link and select **MessageVariable** from the list.

|         | Name   | Type                | Read from Variable                       | Store into Variable          |
|---------|--------|---------------------|------------------------------------------|------------------------------|
| Inputs  | Input  | CustomerApplication | <code>CustomerApplicationVariable</code> | <code>MessageVariable</code> |
|         | Name   | Type                |                                          |                              |
| Outputs | Output | Message             |                                          |                              |

31. Link Final Application Review to Generate Decline and change the Display Name to: `Application is declined`

1. Right-click **Final Application Review** and click **Add a link** from the menu.
2. Click **Generate Decline** to add the link.
3. In the **Properties** view on the **Description** tab, change the **Display Name** to: `Application is declined`

The current path in your diagram looks like the following figure:



32. Set the link condition between Final Application Review and Generate Decline so the process flows to Generate Decline when applicationDecision is false.

If the account reviewer denies the customer application, the application decision attribute is set to `false`. The Generate Decline and Record Declined Application archive the declined application. These services are implemented in a later exercise.

1. Click the link between Final Application Review and Generate Decline.
2. Switch to the **Details** tab in the **Properties** view.
3. In the **Expression Language** field, select **Same as Process (Java)**.
4. For **Expression Type**, click **Java**.
5. Copy the `Final Application Review --> Generate Decline` code snippet from `C:\labfiles\Support\Files\Ex6\AccountVerification_snippets.txt`.

```

// Final Application Review --> Generate Decline

return !CustomerApplicationvariable.getBoolean("applicationDecision");

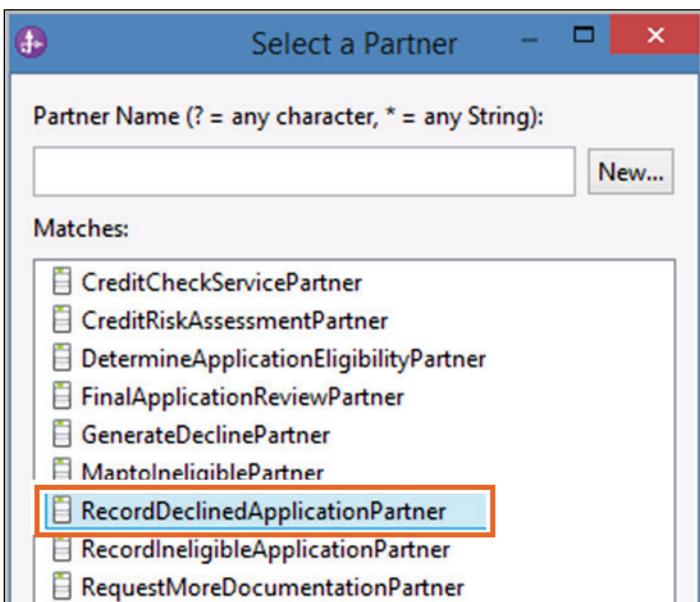
```

6. Paste the snippet in the expression window over the existing text. Alternatively, you can replace the code with the following text:  
`return  
!CustomerApplicationVariable.getBoolean( "applicationDecision" );`

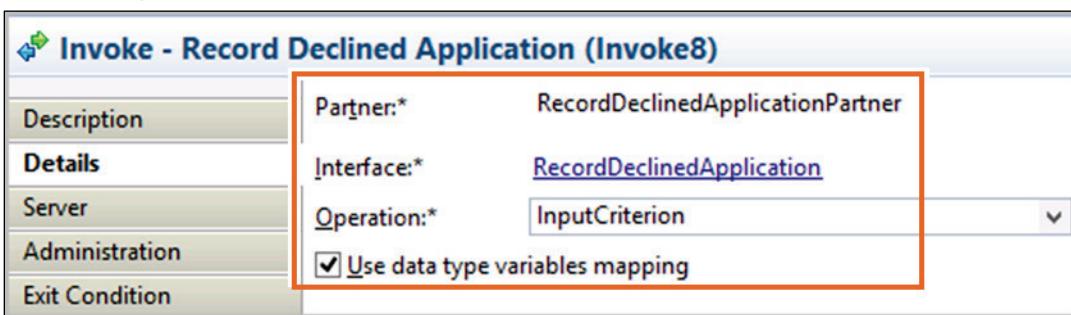
33. Add an invoke activity under Generate Decline with the Display Name: Record Declined Application

The Record Declined Application activity invokes the InputCriterion operation of the RecordDeclinedApplication interface of RecordDeclinedApplicationPartner.

1. In the palette, expand **Basic Actions** and select **Invoke**.
2. Click under **Generate Decline** to add the invoke activity to the diagram.
3. In the **Properties** view on the **Description** tab, change the **Display Name** to: Record Declined Application
4. Switch to the **Details** tab in the **Properties** view.
5. Beside the **Partner** field, click **Browse** and select **RecordDeclinedApplicationPartner** from the “Select a Partner” dialog box.



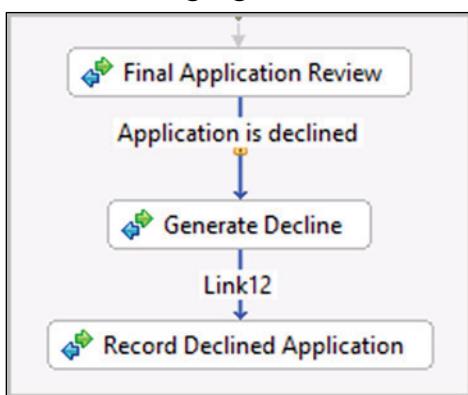
6. Click **OK**.
7. Verify that the **Interface** field is set to `RecordDeclinedApplication`, the **Operation** is set to `InputCriterion`, and the **Use data type variables mapping** check box is selected.



8. In the **Inputs Read from Variable** column, click the **none** link and select **CustomerApplicationVariable** from the list.

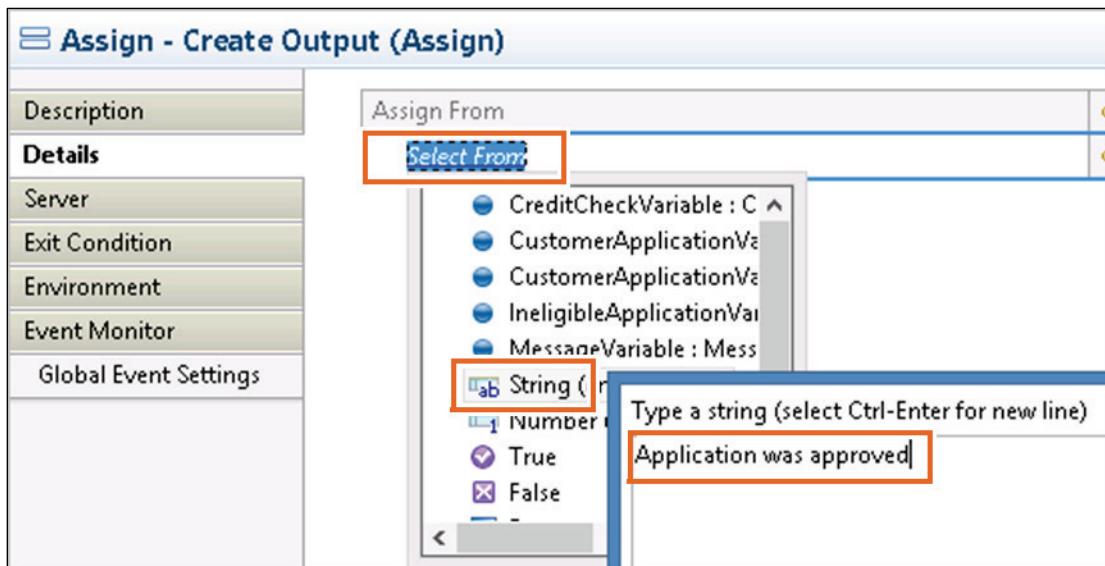
|        | Name  | Type                | Read from Variable          |
|--------|-------|---------------------|-----------------------------|
| Inputs | Input | CustomerApplication | CustomerApplicationVariable |

34. Link Generate Decline to Record Declined Application.
1. Right-click **Generate Decline** and click **Add a link** from the menu.
  2. Click **Record Declined Application** to add the link.
  3. Accept the default link display name. The current path in your diagram looks like the following figure:

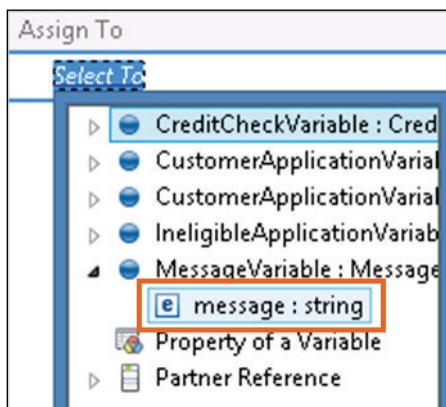


35. Add an assign activity to the right and under Final Application Review with the Display Name: Create Output
- Create Output sets the process output message to: Application was approved. The process returns this message when creditRisk is HIGH or MED, and the person who reviews the application approves it.
1. In the palette, expand **Basic Actions** and select **Assign**.
  2. Click to the right and under **Final Application Review** to add the **Assign** activity to the diagram.
  3. In the **Properties** view on the **Description** tab, change the **Display Name** to: Create Output
  4. Switch to the **Details** tab in the **Properties** view.
  5. In the **Assign From** column, click the **Select From** link.

6. Click **String** and type the following text in the window: Application was approved

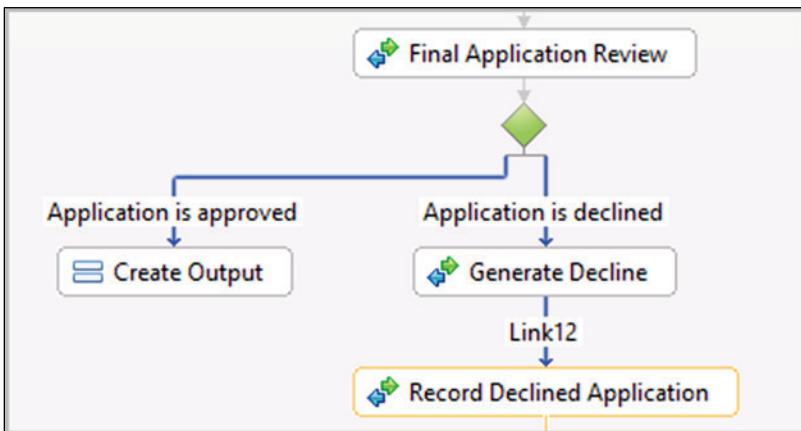


7. Press **Enter**.
8. In the **Assign To** column, click the **Select To** link.
9. Expand **MessageVariable : Message** and select **message : string**.



10. Save your changes. Continue to ignore any errors in the Problems view.
36. Link Final Application Review to Create Output. Change the link Display Name to: Application is approved
1. Right-click **Final Application Review** and click **Add a link** from the menu.
  2. Click **Create Output** to add the link.
  3. In the **Properties** view on the **Description** tab, change the **Display Name** to: Application is approved

4. Save your changes. The current path in your diagram looks like the following figure:



37. Set the link condition between Final Application Review and Create Output so that the process flows to Create Output if applicationDecision is set to true. If the reviewer determines that the customer application deserves to be approved, the application decision attribute is set to true.
1. Select the link between Final Application Review and Create Output.
  2. Switch to the **Details** tab in the **Properties** view.
  3. In the **Expression Language** field, select **Same as Process (Java)** from the list.
  4. For **Expression Type**, click **Java**.
  5. Copy the Final Application Review --> Create Output code snippet from C:\labfiles\Support Files\Ex6\AccountVerification\_snippets.txt.
- ```

// Final Application Review --> Create output
-----
return CustomerApplicationvariable.getBoolean("applicationDecision");
  
```
6. Paste the snippet in the expression window over the existing text. Alternatively, you can replace the code with the following text:

```

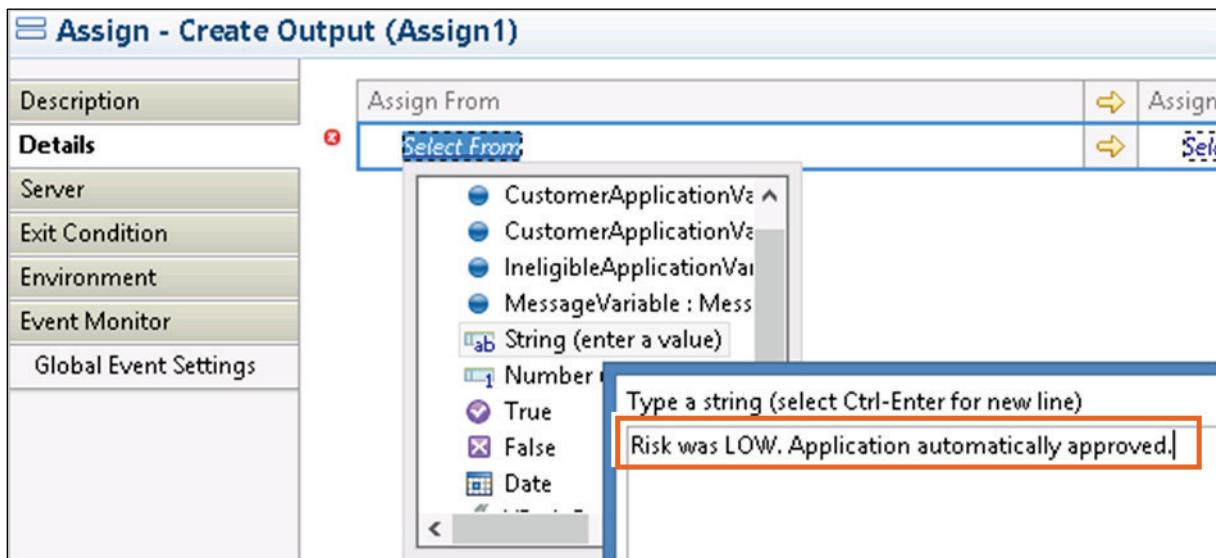
return
CustomerApplicationVariable.getBoolean( "applicationDecision" );
  
```

 7. Save your changes.

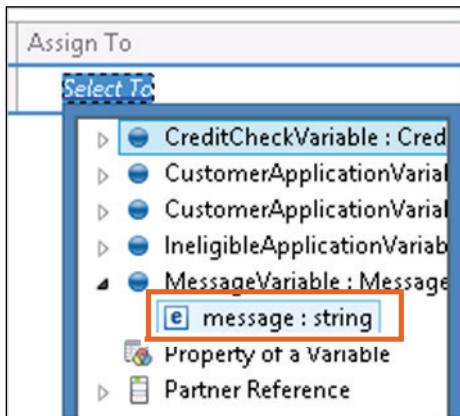
38. Add an assign activity to the right and under Credit Risk Assessment with the Display Name: Create Output

This Create Output activity sets the process output message to: Risk was LOW. Application automatically approved. The process returns this message when creditRisk is LOW, and the application flows directly from Credit Risk Assessment to Account Verification Reply.

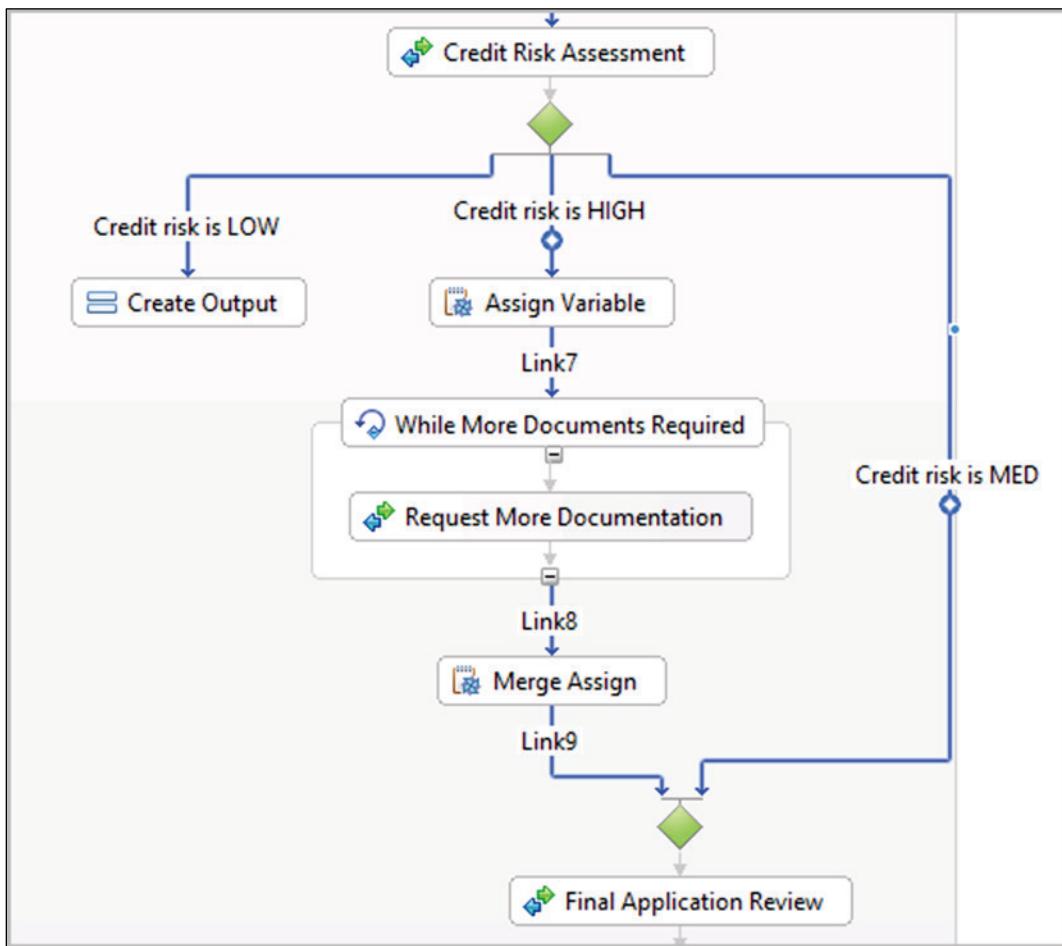
1. In the palette, expand **Basic Actions** and select **Assign**.
2. Click under and to the right of **Credit Risk Assessment** to add the activity to the diagram.
3. On the **Description** tab in the **Properties** view, change the **Display Name** to: Create Output
4. Switch to the **Details** tab in the **Properties** view.
5. In the **Assign From** column, click the **Select From** link.
6. Click **String** and type the following text in the window: Risk was LOW. Application automatically approved.



7. Press **Enter**.
8. In the **Assign To** column, click the **Select To** link.
9. Expand **MessageVariable : Message** and select **message : string**.



10. Save your changes. Continue to ignore any errors in the **Problems** view.
39. Link Credit Risk Assessment to Create Output and change the link Display Name to: credit risk is LOW
 1. Right-click **Credit Risk Assessment** and click **Add a link** from the menu.
 2. Click **Create Output** to add the link.
 3. In the **Properties** view on the **Description** tab, change the **Display Name** to: Credit risk is LOW
 4. Save your changes. The alignment might be slightly different from the following figure:



40. You will set the link condition between Credit Risk Assessment and Create Output so that the process flows directly to Create Output if creditRisk is LOW.

If the credit risk is HIGH, customer applications flow through Credit Risk Assessment to Assign Variable. If the credit risk is MED (short for “medium”), data flows through Credit Risk Assessment to Final Application Review and skips the gathering of more documentation. If the credit risk is LOW, data flows through Credit Risk Assessment to Create Output and then to Account Verification Reply.

1. Click the link between Credit Risk Assessment and Create Output.
2. Switch to the **Details** tab in the **Properties** view.
3. In the **Expression Language** field, select **Same as Process (Java)**.
4. For **Expression Type**, click **Java**.
5. Copy the **Credit Risk Assessment --> Create Output** code snippet from C:\labfiles\Support Files\Ex6\AccountVerification_snippets.txt.

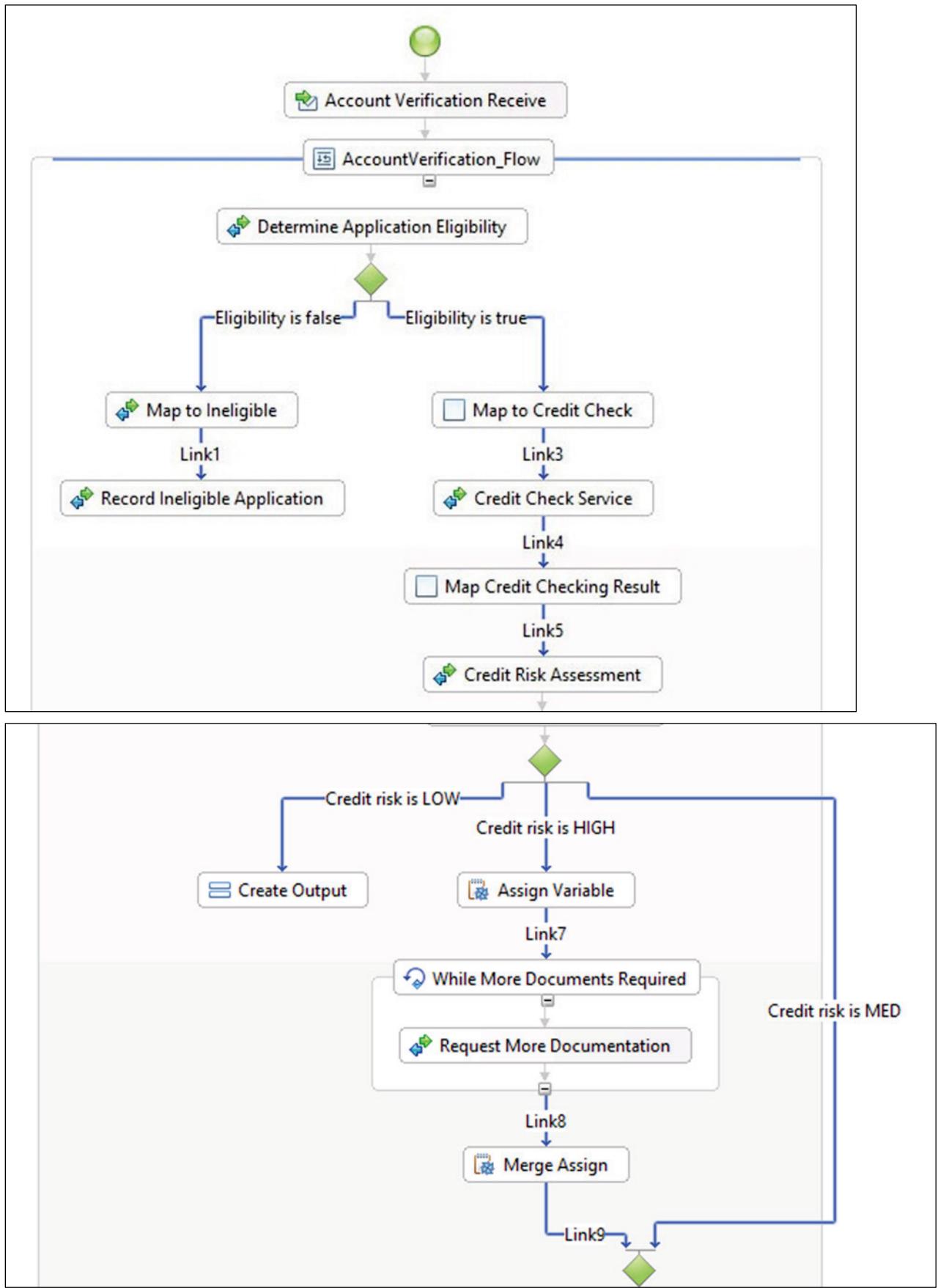
```
//-----
// Credit Risk Assessment --> Create Output
//-----
return CustomerApplicationvariable.getString("creditRisk").equals("LOW");
```

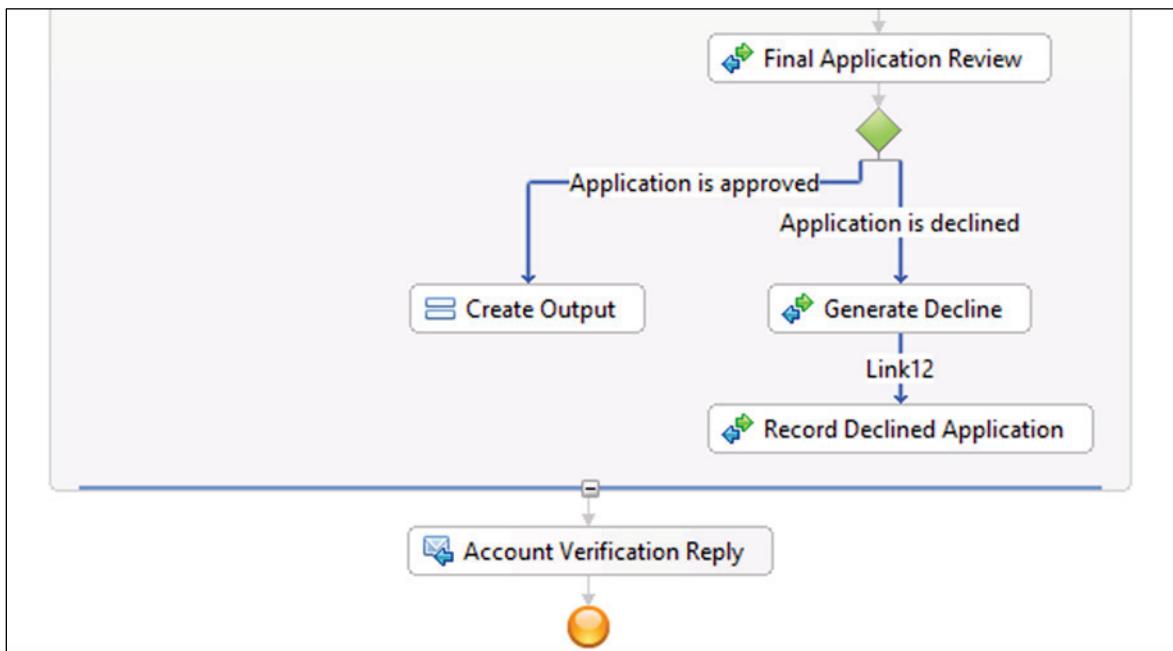
6. Paste the snippet in the expression window over the existing text. Alternatively, you can replace the code with the following text:

```
return  
CustomerApplicationVariable.getString( "creditRisk" ).equals( "LOW" );
```

7. Save your changes.
8. Verify that the **Problems** view has no errors.
9. Save your changes. Your process looks like the following diagram:

Note: The diagram spans more than one page for clarity. You can also open the following .PNG image of the completed business process in your lab environment: C:\labfiles\Support Files\EX6\AccountVerification_Complete.png





10. Close Windows Explorer and C:\labfiles\Support_Files\Ex6\
AccountVerification_snippets.txt.

Results:

In this exercise, you used IBM Integration Designer to implement basic and structured activities in a business process.

Unit 10 Business process handlers, runtime behavior, and clients

IBM Training



Business process handlers, runtime behavior, and clients

IBM Business Process Manager V8.6

© Copyright IBM Corporation 2018
Course materials may not be reproduced in whole or in part without the written permission of IBM.

Unit objectives

- List and describe the available handlers and error-processing activities
- Describe the runtime behavior of business processes
- Describe the administrative options and types of client access that are available for business processes

Unit objectives

This unit describes the various types of activities in BPEL, including basic and structured activities.

Topics

- WS-BPEL handlers and error processing activities
- Runtime aspects of BPEL processes

WS-BPEL handlers and error processing activities

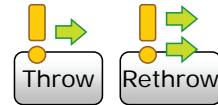
Business process handlers, runtime behavior, and clients

© Copyright IBM Corporation 2018

WS-BPEL handlers and error processing activities

Handlers and error processing activities: Throw and rethrow

- A throw activity signals a condition that the logic cannot handle (a fault)
- Faults are matched:
 - By fault name alone (the parameter might not match); parameter can contain added fault information
 - By optional parameter alone (not suggested)
 - Or both (best choice for a good match)
- Throws can raise a built-in or user-defined fault
 - The engine throws built-in faults to signal low-level failures in the system
 - BPEL-defined standard faults for common failures (such as joinFailure)
 - User-defined faults that a business partner or process throws
- The rethrow activity takes the current fault in a fault handler and raises it to the next enclosing scope
 - Rethrow raises a new fault with the same name and parameters as the fault that the containing handler caught
 - Indicates that the error state remains
 - Rethrow is available only in fault handlers



Handlers and error processing activities: Throw and rethrow

A throw activity in a business process can throw any fault, including standard faults, but the intended usage pattern is to throw business faults. A throw activity throws an exception that can be caught and handled within the business process. If a throw activity throws a fault and it is not handled within a process with a request/response interface, either business or standard fault, it is returned as a runtime exception to the process caller.

You cannot return a business or standard fault with a throw activity. You must use a reply activity to return a business fault to the process client. A reply activity can return a business fault that is defined only on the interface that the process implements. This business fault is returned to the process caller; you cannot catch and handle it within the process.

A rethrow activity can be used in a fault handler to rethrow the fault to the next enclosing scope. This action might be useful when you want to do some fault handling on the current scope, such as triggering specific compensation handlers, and still want to make the enclosing scopes aware of this issue.

You can also use a rethrow activity when the current fault handler cannot handle the fault and wants to propagate it to an outer-scoped fault handler. In the absence of a rethrow activity, a fault that uses a throw activity to propagate to a higher level is a new fault. When a rethrow activity is invoked, the fault is the same instance. The rethrow activity is available only within a fault handler because only an existing fault can be rethrown. A rethrow activity is especially useful in a catch-all fault handler because the current fault data cannot be easily retrieved when a throw activity is used.

The Web Services Business Process Execution Language (WS-BPEL) specification defines standard fault types for common system failures. You can add these built-in fault types to your process definition. These faults are available only within BPEL processes, and they do not have an equivalent in the SCA programming model.

Handlers and error processing activities: Fault handler

- A fault handler is a collection of specific activities that run when a fault (expected error) is thrown on the activity or scope 
- Fault handlers use “catch” and “catch all” elements to handle error conditions in a process
 - “Catch” elements process specific faults
 - “Catch all” elements process faults that a “catch” element does not intercept
- A scope can have multiple fault handlers
 - Different kinds of faults can have different fault-handling activities



Business process handlers, runtime behavior, and clients

© Copyright IBM Corporation 2018

Handlers and error processing activities: Fault handler

Fault handlers are designed to detect and signal a failure in the execution of a business process. They are designed to catch business exceptions such as a customer who requests an item that is no longer available. The fault handler can catch these faults and either handle the exception and continue processing or rethrow the fault. A fault handler is associated with an invoke activity, a scope of activities, or a process. When a handler catches a fault, any activities in progress are terminated and execution switches to the fault handler.

Fault handlers can be nested like scopes. Each scope can contain its own fault handler. If a fault is not caught at the current scope, it propagates up to the next scope, and the fault handler for that scope catches it. If the fault reaches the uppermost scope of the business process and is not handled, the entire business process fails. Within a fault handler, you can also define compensate activities that trigger a compensation handler.

When a fault is thrown

- A fault means that processing in the current scope cannot continue
 - All active work in the scope fails
- If a matching fault handler is defined in the scope:
 - The scope is in a FAILED state, but the enclosing scope continues with normal processing
 - The fault is handled as specified in the “catch” block
- If a handler is defined but does not match the fault, or if no fault handler is defined:
 - The scope is in a FAILED state, and the fault is rethrown to the next enclosing scope
 - If a fault escalates to the top-level scope of the process, it is considered unhandled, and the entire business process is in a FAILED state
- If a fault is not caught on the enclosing scope or in a fault handler, the “continue on error” setting determines how the process proceeds
 - “No”: The process is STOPPED and a work item is created for the process administrator
 - “Yes”: The fault is rethrown and if it reaches outermost scope, the process is in a FAILED state

When a fault is thrown

A fault handler or fault link can catch a specific fault name, fault type, or both. When a fault occurs, the Business Flow Manager uses the following rules to match the fault with a fault handler or fault link on the enclosing scope, or on the activity where the fault occurred.

- If an invoke activity without a fault handler or any other basic activity is the source of one or more fault links, the Business Flow Manager tries to find a matching fault link. If a fault link is not available, it then tries to find a matching fault handler on the enclosing scope.
- If an invoke activity or a scope with one or more fault handlers is the source of one or more fault links, the Business Flow Manager tries to find a matching fault handler. If a fault handler is not available, it runs the default fault handler and then tries to find a matching fault link. If a matching fault link is not available, it tries to find a matching fault handler on the enclosing scope.
- If the fault does not have any associated fault data, the Business Flow Manager uses a fault handler or fault link with the matching fault name. If a fault handler or fault link is not found, it uses the catch-all fault handler or fault link if one is available. A fault handler or fault link with a fault variable that is defined cannot catch a fault that does not have any data.

- If the fault has fault data that is associated, the Business Flow Manager uses a fault handler or fault link with the matching fault name. It also uses a fault variable with a type that matches the type of the fault data. If a fault handler or fault link is not found that matches the name and fault data type, it uses a fault handler or fault link without a fault name. It also uses a fault variable with a type that matches the type of the fault data. If a suitable fault handler or fault link cannot be found, it uses the catch-all fault handler or fault link if one is available. If a fault handler or fault link does not have a fault variable that is defined, it cannot catch a fault that has data.

If a fault is raised that does not match any of the fault handler or fault link definitions, the default fault handler is started. The default fault handler is not specified explicitly. The default fault handler runs all of the available compensation handlers for the immediately enclosing scopes in the reverse order of completion of the corresponding scopes. If the scope is the source of one or more fault links, the Business Flow Manager then tries to find a matching fault link. If a matching fault link is not available or the scope is not the source of any fault links, the default fault handler rethrows the fault to the next level. The next level is the enclosing scope of the process. On this next level, the Business Flow Manager again tries to match the fault to the fault handlers or fault links that are available.

If any of the specific fault handlers or fault links do not catch the fault – for example, the catch-all fault handler or catch-all fault link – the fault reaches the process scope. The process ends in the FAILED state. Even if a fault handler catches the fault on the process scope and handles it, the process still ends in the FAILED state.

When you define a business process, you can specify what happens when an unexpected fault is raised and a fault handler is not defined for that fault. You can use the “continue on error” setting when you define your process to specify that it is to stop where the fault occurs.

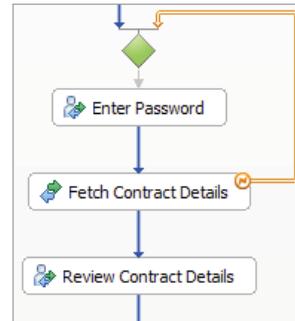
For most activities, the continue on error behavior is the same as for the process. You can specify the continue on error behavior explicitly for invoke, Java snippet, custom, and human task activities. By default, the continue on error behavior of these activities is also the same as for the process.

If an unexpected fault is detected, fault handling of the activity is started. If the continue on error setting is set to “yes,” then the standard fault handling is applied. The continue on error setting for the activity or the process might be set to “no.” In this case, if a fault handler on the immediately enclosing scope or a fault link that leaves this activity does not handle the fault, then the activity stops. If a catch-all fault link or fault handler is defined on the immediately enclosing scope, the value of the continue on error setting has no effect. The fault is always handled, and the activity is never stopped.

For activities that stop because of an unexpected fault, you can use the stop reason property of the activity, which determines the cause of the fault and the actions that you can take to repair it.

Error processing: Fault links

- Fault links are run after a fault occurs in the source activity of the link
 - When a fault occurs and is caught, only the fault link is navigated
 - The fault path can merge back into the regular flow
 - Evaluation order of multiple fault links can be specified
- Fault catching rules are the same as for fault handlers
 - A fault name and fault variable to catch can be specified (Catch), or all faults can be caught (Catch All)
- Fault links are available in generalized flows



Fault Link - Link6

<input checked="" type="radio"/> Catch	<input type="radio"/> Catch All
Description	Fault Type:
	<input type="radio"/> Built-in <input checked="" type="radio"/> User-defined
Details	Fault Name:
	{http://BetterFinancials/Processes/ProvideLoanInformation/ProvideLoanInformationInterface}badPasswordFault
	Fault Variable:
	(none) Browse...

Business process handlers, runtime behavior, and clients © Copyright IBM Corporation 2018

Error processing: Fault links

When faults occur in your business process, fault handlers are typically engaged to deal with the fault. The generalized flow activity offers a simplified fault handling procedure. From any scope or basic activity (excluding the throw and rethrow activities), you can add one or more “fault links.” If a specified fault occurs while an activity is running, the fault link in the activity is followed. You can define “catch” fault links for various conditions, or you can create a “catch-all” fault link. The catch-all fault link must be followed when any fault occurs that a catch fault link does not cover. If multiple fault links are modeled for the same activity, best match decides which fault link to follow. The fault that catches rules is the same as for fault handlers. No more than one fault link can catch an occurred fault.

You can terminate the fault link at any activity within the generalized flow. For example, you can choose to direct the fault link to a terminate activity to have the business process stop if a fault occurs. You might want the business process to skip the rest of the steps in the generalized flow and exit to the next activity. In this case, you would terminate the fault link at the last activity in the generalized flow. You can also use a fault link to create a cycle and loop back to a previous activity.

Handlers and error processing activities: Compensate

- Compensation activities allow business processes to define activities that place the business back into a balanced state
 - A balanced state is an outcome acceptable to both the client and the business
 - Only scopes or activities that completed normally can be compensated
- The runtime behavior and objective of compensation differ:
 - Compensation in a long-running process attempts to restore balance after one or more transactions are committed
 - In a microflow, the entire process runs within a single transaction
 - To set up compensation for a microflow, you store the properties for the invoke activities
 - If the process cannot be committed and must be rolled back, the original data can then be restored
 - If a microflow fails, the runtime engine crawls back through the invoke activities in reverse order, restoring each to its previous state
 - To create compensation logic for an invoke activity in a microflow, you must define a location to store the process state details, called a compensation sphere
- The compensate activity is available in a fault handler, a compensation handler, a collaboration scope, or a generalized flow activity

Handlers and error processing activities: Compensate

The compensate activity can be used in a fault handler or a compensation handler. The compensate activity points to a specific compensation handler, which contains the steps necessary to restore a process to a balanced state. In addition, compensation activities can be added to a generalized flow activity or a collaboration scope.

Previously, compensate activities were used only in compensation handlers and fault handlers (as specified by the BPEL specification). When a compensate activity was added to a generalized flow or collaboration scope, the entire generalized flow or collaboration scope was compensated; you could not specify a specific compensation target in these activities.

Compensation has at times been described as a means of undoing an action, but this description is not entirely accurate. More specifically, it is a service that runs when a state is reached in your process that you deem undesirable. The goal is not always to return to a previous condition, but instead to maintain a balanced and consistent state and to compensate any committed operations that conflict with this state. Two types of compensation exist: business compensation and technical compensation.

- **Business compensation:** This type of compensation is used in a transactional process where an operation is already committed and cannot be reversed. Business compensation is another operation that, when run, creates a balanced state where both business partners are satisfied.

For example, suppose that something goes wrong at any time during a typical business transaction. It is a simple matter of replacing the object on the store shelf, and halting all communication between the purchaser and the vendor. However, if the transaction is committed (money is exchanged and a receipt issued), then canceling it is not possible. You cannot return the object to the shelf. A different procedure (a refund) must take place to return the conditions to a balanced state. The operations that already took place must be compensated to return to a situation in which both partners are satisfied. It is not necessarily the same state that existed before (for example, if the customer paid in cash, the customer can receive a store credit in return). Nonetheless, it is one that is balanced and consistent.

- **Technical compensation:** Technical compensation is used in transactions that fail before they are committed, when one of the operations cannot be reversed. For example, imagine that a customer requests that an item is personalized in some way. The vendor complies, but before money is exchanged, something unexpected happens, and the transaction is canceled. The object cannot be returned to the shelf; another procedure must run to consider the personalization that took place. In another example, imagine that in your process, one of the activities within a transaction sent out an email, but the transaction was canceled before it was committed. You cannot undo the sending of an email, so you must compensate in some other way.

The two ways in which you can compensate business processes are:

- **Compensation pairs:** Compensation pairs are the original properties of each of the individual parts of a business process. These properties are saved so that they can be restored if the process cannot be committed and must be rolled back. The original status of the activity is stored in an operation, and its value is stored in a variable. If IBM extensions are not enabled for this process, then you cannot use compensation pairs and must use a compensation handler.
- **Compensation handler:** A compensation handler is a series of isolated activities. The activities in the handler run only when a fault is thrown, and after the parent activity is already committed. The goal of a compensation handler is to return a failed process to a balanced state.

You cannot use a compensation handler with a microflow. Because microflows run within a single transaction, you must use compensation pairs to store the original properties of each invoke activity if the process fails.

In a long-running process, you can use either of these options. If the compensation characteristics of each activity are fairly simple (compensation can be achieved in a single step), then consider the use of compensation pairs. However, if you require compensation that uses more complicated logic, assign a compensation handler to each activity.

IBM Training **IBM**

Handlers and error processing activities: Compensation handlers

- Compensation handlers contain actions that reverse operations that are designed to handle unexpected system problems and bring the process back to a known, stable state
- Compensation handlers can be added to invoke, scope, generalized flow, collaboration scope, or human task activities

```

graph TD
    Indicator --> BillCustomer
    BillCustomer --> CompleteOrder
    CompleteOrder --> ShipOrder
    ShipOrder -- Fault --> CatchAll
    CatchAll --> RestockItems
    RestockItems --> CompBillCustomer
    CompBillCustomer --> RefundCustomer
    RefundCustomer --> CompCompleteOrder
    subgraph CompensationHandlers [Compensation handlers]
        RefundCustomer
        RestockItems
        CompBillCustomer
    end

```

Business process handlers, runtime behavior, and clients © Copyright IBM Corporation 2018

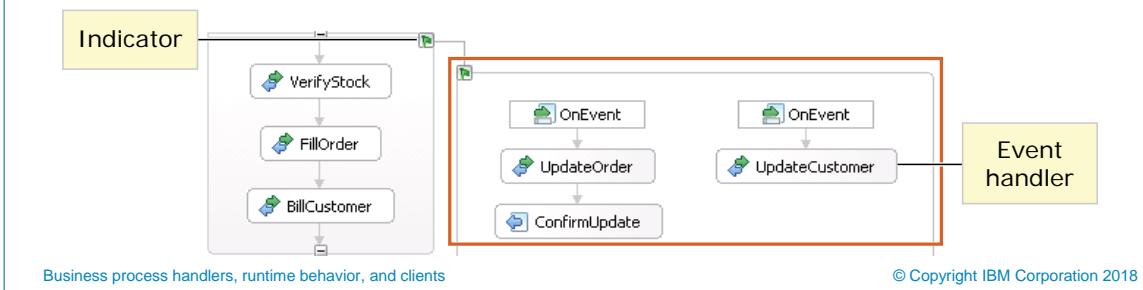
Handlers and error processing activities: Compensation handlers

In this example, if an order fails to ship for some reason, a fault is thrown and is caught by using a “catch-all” block in a fault handler. The compensation activity in the fault handler then calls the appropriate compensation handler. This handler restocks the items on the order. The next compensation handler is then called, which would handle refunding the customer. Although the compensation handlers in this example are attached to individual activities, they can be attached to scopes to include multiple activities. During compensation:

- All variables and partner links are accessible because all variable values are persisted in the database when the completed activity commits. Each completed activity creates a row in the database that contains a snapshot of the process variable values.
- The values (such as a confirmation number) can be used in the compensation handler to “undo” the completed activity.

Handlers and error processing activities: Event handlers

- **Event handlers** accept outside requests for a particular scope
 - **OnEvent** elements allow the process to consume messages from the client
 - **Timeout** elements allow the process to work on a task after a specified length of time (a timer starts when a scope is entered)
- Any type of processing can occur in the event handler (sequence or parallel)
- Requests can be one-way or request/response, and events can be repeated until the activity or scope is completed
- Restrictions can be set to limit who can send events
- Correlation sets must be enabled on event handlers to direct events to the correct business process instances



Handlers and error processing activities: Event handlers

Event handlers were originally defined under the BPEL V1.1 specification. Event handlers are available only for a long-running business process. They accept outside requests from external clients into the business process.

In this example, while the specified scope is active, a customer can add or remove products from the order or update the customer information.

You can define and configure an event handler either on individual scopes within the process, or for the process in its entirety. An event handler can be used to process requests while the process is running. To process the event, the handler has access to business process state information such as variables.

An event handler has two parts. The first part defines the conditions under which an event handler is invoked, and it defines the type of the event handler. You have the following two choices:

- OnEvent element: Use this element to create a control path and specify the operation that causes this path to be followed.
- Timeout element: Use this element to create a control path that is followed when a specified time is either reached or elapsed. This element is used on a single path, and is configured to specify a specific date or time. When the process is running, this path is chosen unless input is received within this time period, or by the specified date.

The second part of an event handler defines the implementation of the event handler; specifically, the action that the event handler is to take when it runs. To specify this action, populate that handler with the necessary process activities from the palette.

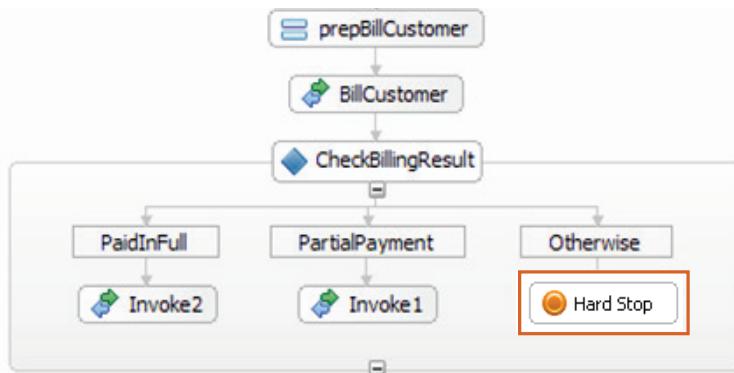
Event handlers use correlation to match requests to the correct process instance. When a handler receives an event, a set of steps that is defined in the handler runs in a separate thread from the business process. When the event handler activities are complete, the thread stops.

Each event handler is enabled when the process or scope that it is associated with starts. Each event handler is disabled when the process or scope that it is associated with ends. Multiple instances of each event handler can be started while the event handler is enabled. If an event comes in before this event handler is available (when the scope is reached), the business process engine holds that event and waits until that scope is reached. If the event is received after the scope is ended, a runtime exception is generated, indicating that the event cannot be handled.

The event activity runs on its own thread, created by the process container upon dispatching the event.

Handlers and error processing activities: Terminate

- A **terminate** activity ends processing of the business process instance (a hard stop) or the structured activity
 - Can be used to allow the administrator to make repairs
- All execution ends, including parallel execution paths
- Compensation of terminated activities does not occur



Business process handlers, runtime behavior, and clients

© Copyright IBM Corporation 2018

Handlers and error processing activities: Terminate

The terminate activity is the equivalent of pulling the plug on an electrical device. It is a hard stop, signifying that the business process must be halted. When it is used, all activities that are currently active are halted without any fault handling or compensation behavior.

Runtime aspects of BPEL processes

Business process handlers, runtime behavior, and clients

© Copyright IBM Corporation 2018

Runtime aspects of BPEL processes

Business process runtime support

- Service integration buses support process communication and navigation
 - Java Message Service (JMS) messaging resources for your business processes are created during application deployment
 - Process transactions can also be triggered by using the work manager
 - In work-manager-based navigation, work-manager threads are used to reduce JMS use and improve performance
- Process state and activity-related information can be stored in a database for persistence, recovery, and data queries
 - Database is called BPEDB by default
 - Database must be created manually in IBM Process Server, by using a supported production database, such as DB2 or Oracle
 - Database creation scripts are available at
`<WPS_Install>\dbscripts\ProcessChoreographer`
 - Database creation is automatic in IBM Integration Designer
 - DB2 Express is the data store for IBM Integration Designer
- If your business process uses human tasks, you must also install the human task container

Business process runtime support

The business process container is the resource that is responsible for running the business processes. It is a Java EE application front end that provides the APIs that you can use to create robust clients. The business process container can be installed by default when you install IBM Process Server. It is automatically installed with IBM Integration Designer as part of the test environment.

If you elect not to install the container when you install a stand-alone IBM Process Server, you can configure it later by using the IBM Process Server administrative console. The administrative console includes a wizard that guides you through the container configuration options such as the database type, messaging resources, and security parameters.

In IBM Integration Designer, the container is configured to use a Derby database by default, which is adequate for development and testing. A scalable, enterprise database such as DB2 is suggested for production environments. DB2 on z/OS, Oracle, and a number of other database products are supported. Scripts are provided to set up the database with the tables that the business process container needs. The data store holds the business process templates, along with business process instance and state information for the container. Messaging resources for business processes are enabled and configured in the environment as well.

A lightweight navigation style, called work-manager-based navigation, is also available. In work-manager-based navigation, work-manager threads are used. The quality of service is the same as in JMS navigation. Communication to the database is reduced, which results in performance improvements. For more information about work-manager-based transactions in business processes, see the topic “Improving the performance of business process navigation” in the product documentation.

Business process templates and instances

- Process templates describe the business process model
 - The Business Flow Manager uses the template to create instances of the business process at run time
 - Process templates are deployed and installed on IBM Process Server
- Process instances are entities that exist at run time
 - An instance represents one running business process, its specific data, and its state
 - The Business Flow Manager can run multiple process instances at the same time
- Creation of a template version is supported by using the template's valid-from date
 - The valid-from date is used to decide which process template to use when creating a process instance
 - When the instance is created, it runs against that version of the template
 - If a new template is deployed, in-flight process instances can be upgraded
- Process templates and instances are persisted to a database
 - In IBM Integration Designer, the database is in DB2 Express
 - For production environments, database creation scripts are provided per platform

Business process templates and instances

By using the process instance migration, you can migrate in-flight process instances to the new version at run time. To create a version of your process, you can define a process migration specification, which provides process instance migrations at run time. You deploy and install the new process version (with a new validFrom date) and the process migration specification to your runtime environment. An administrator can then migrate existing instances of the process in the runtime environment to the new version of the process. The processes can be migrated in two ways: the BPEL Process Choreographer Explorer application for select business processes, or a script for batch migration.

Of critical importance: The two versions must have the same name and namespace; but with different valid-from dates, the correlation sets for the process versions must be the same, and the interfaces must be the same. By using the process instance migration tools, you can update versions of running instances of processes in a late-binding situation. With early binding, a client is hardwired to a particular process version.

For an example of process instance migration, see:

<http://publib.boulder.ibm.com/bpcsamp/index.html>

Microflow and long-running processes at run time

- The runtime behavior of a business process depends upon whether it is long-running or microflow (non-interruptible)

	Microflow (non-interruptible)	Long-running (interruptible)
Transactions	One transaction for the entire process	Transaction boundaries can be set between activities
Persistence	None	Information about process activity and state is persisted to a database
Crash recovery	None (execution is transient; if the server crashes, the process state is lost)	Process can resume from the last checkpoint
Parallelism	None: strictly single-threaded	Activities can run in parallel
Interruptible	No	Yes
Asynchronous	No	Yes
Correlation	No	Yes
Handlers	No support for fault, event, or compensation handlers	Handlers are supported

Business process handlers, runtime behavior, and clients

© Copyright IBM Corporation 2018

Microflow and long-running processes at run time

In this course, transactions and persistence are not covered in detail. However, it is still necessary to point out these key differences between microflow and long-running business processes. While they are both considered business processes, a microflow is contained within a single transaction, where a long-running process can be multi-transactional and can run over an extended period. If your process requires more than one transaction, it must be a long-running process.

If your process can be designed either way, consider the following options:

- If your process must stop at any point and wait for external input, in the form of either an event or a human task, then you must use a long-running process. Microflows are not interruptible.
- If you do not have IBM extensions that are enabled for this process, then you cannot make it a microflow. A microflow is an IBM enhancement of the BPEL programming language. If these extensions are disabled, a microflow is not an option.
- If you have a short series of steps to model and want them to run quickly in the runtime environment, then use a microflow. Microflows can also be used as subprocesses in a larger business process.

- If you have elements in your process that you would like to run in parallel, use a long-running process. Keep in mind that to work properly, each of the parallel paths must run within its own transaction, and its transactional behavior must be set to be either “commit before” or “requires own.”

If IBM extensions are disabled, the **Process is long-running** check box is not displayed on the Details tab of a long-running process. However, you can enable the extensions by clearing the **Disable IBM Process Server BPEL extensions** check box on the Details tab. When you do so, you are warned that this step cannot be undone. If you proceed, then the **Process is long-running** check box is displayed on the **Details** tab. It can be cleared to convert the process to a microflow.

Clients for business processes

- Several clients are available for working with process instances
- Business Process Choreographer Explorer:
 - Is a customizable application for running basic administrative tasks on business processes
 - Reporting is a customizable application for reporting on business processes
- Generic EJB, JMS, and web service APIs are used to create clients that make direct calls to the business process container
 - Can query business process instances
 - Can interact with instances (start, send events, and invoke activities)
 - APIs are documented in the IBM Integration Designer product documentation
- Clients interact with processes by using different service interfaces and bindings:
 - SCA, JMS, and web services
 - Clients start processes (through a receive or receive choice activity) or send events

Clients for business processes

The BPEL Process Choreographer Explorer reporting component requires the Common Event Infrastructure EVENT database to compile statistics.

The Business Process Choreographer web services API provides two separate web service interfaces:

- The Business Flow Manager API allows client applications to interact with microflows and long-running processes, for example:
 - Create process templates and process instances
 - Claim existing processes
 - Query a process by its ID
- The Human Task Manager API allows client applications to:
 - Create and start tasks
 - Claim existing tasks
 - Complete tasks
 - Query a task by its ID
 - Query a collection of tasks

Client applications can use either or both of the web service interfaces.

The Enterprise JavaBeans (EJB) APIs are provided as two stateless session enterprise beans. Business process applications and task applications access the appropriate session enterprise bean through the home interface of the bean.

The `BusinessFlowManagerService` interface provides the methods for business process applications, and the `HumanTaskManagerService` interface provides the methods for task-based applications. The application can be any Java application, including another Enterprise JavaBeans (EJB) application.

With these Enterprise JavaBeans (EJB) APIs, you can create client applications to do the following tasks:

- Manage the lifecycle of processes and tasks, from starting them to deleting them when they complete
- Repair activities and processes
- Manage and distribute the workload over members of a workgroup

The EJB APIs are provided as two stateless session enterprise beans:

- The `BusinessFlowManagerService` interface provides the methods for business process applications.
- The `HumanTaskManagerService` interface provides the methods for task-based applications.

For more information about the EJB APIs, see the Javadoc in the `com.ibm.bpe.api` package and the `com.ibm.task.api` package.

You can use the generic JMS client interface (referred to as the “JMS API”) to develop client applications that asynchronously access business processes that run in the Business Process Choreographer environment. The JMS API allows client applications to asynchronously interact with microflows and long-running processes. The JMS API exposes the same interface as the web services API, with the following exceptions:

- With the web services API, the call operation can be used to invoke microflows only.
- However, by using the JMS API, the call operation can be used to invoke both microflows and long-running processes.

IBM Training IBM

Business Process Choreographer Explorer client

All Tasks

Use this page to work with task instances for which you have access rights. [\[i\]](#)

Work on	Release	Transfer	Start	Change Business Category	Refresh
<input type="checkbox"/> Priority ▼	Task Name ▼	State ▼	Kind ▼	Owner ▼	
<input type="checkbox"/> 5	Request More Documentation	Ready	To-do Task		

Items found: 1 Items selected: 0 << Page 1 of 1 >>

- Business Process Choreographer Explorer is included for basic process administration
- Business Explorer is built with reusable JavaServer Faces (JSF) components

Business process handlers, runtime behavior, and clients © Copyright IBM Corporation 2018

Business Process Choreographer Explorer client

Business Process Choreographer Explorer is included for basic process administration, as follows:

- View installed process templates
- View, start, terminate, delete, compensate, suspend (including “suspend until” and “suspend for”), try process instances again, transfer ownership

Business Process Choreographer Explorer is built with reusable JavaServer Faces (JSF) components, as follows:

- Starting point for customized administrative clients
- Ability to create customized views

The Business Process Choreographer Explorer client provides administrative capabilities for business processes and human tasks. It is installed when you configure the business process container. You can do basic functions such as starting, terminating, and deleting instances. You can compensate activities that fail within the business process, and you can try them again. Security can be enabled for this application to limit who can work with or administer processes and tasks.

Business Process Choreographer Explorer is based on JavaServer Faces, so many of the views are implemented with JSF tags or components and can be reused in your own custom applications and administrative clients. With a little effort towards understanding the JSF tags, you can build your own clients that extend the Business Process Choreographer Explorer.

The Business Process Choreographer Explorer includes support for selective deletion of completed process instances and their associated data from the Business Process Choreographer runtime database.

Deletion criteria include:

- Process instances of a particular process template
- Process instances that a particular user initiates
- Process instances that are finished before a date or time
- Process instances in a particular state – that is, finished, terminated, or failed

You can also use the Business Process Choreographer Explorer to suspend process instances for an amount of time or until a specified time and then automatically resume them. For example, if a client asks to put an order on hold, you can suspend the process for seven days, or until a specific date and time.

Suspending a business process instance means that the navigation of the process instance is put on hold. If you suspend a process instance, you can specify a duration or point in time when the process instance resumes automatically. This capability is exposed through all renderings of the generic BFM interface, that is, to JMS clients, web services clients, and EJB clients. Suspension of a process applies to long-running, top-level processes, and is propagated to all child processes.

You can also use the Business Process Choreographer Explorer to transfer process ownership. For example, if the owner of a process is no longer with your company, you can use this feature of the Business Process Choreographer Explorer to transfer process ownership to another employee.

In the IBM Integration Designer test environment, the Business Process Choreographer Explorer URL is: <http://localhost:9080/bpc>

Runtime process dynamicity (1 of 2)

- Scopes and collaboration scopes support dynamic modification by using the process widget in Business Space or Business Process Choreographer Explorer
 - Provides greater flexibility and allows processes to adapt to changing situations
 - Supports dynamic human workflow scenarios and case handling scenarios
- Dynamicity allows business users and solution administrators to override the navigation of a long-running process
 - Jump forward and backward between activities in a running process
 - Skip specific activities within a running process
 - Incorporate process-relevant data changes at run time

Skip Activities

Use this page to skip activities and, optionally, to view and modify the variables

[i]

Skip	Set Variables	Cancel	←	Dynamically skip activities and set variables
Process Instance Name	LoanApp2			
Description				
State	Running			
Activity Names	<input type="checkbox"/> Receive <input type="checkbox"/> CopyInput <input type="checkbox"/> CheckAutoApproval			

Runtime process dynamicity (1 of 2)

Dynamic modification includes the ability to modify process data at run time. You can modify business information, such as customer name and address. CEI events can be generated during dynamic modification for auditing purposes.

Runtime process dynamicity (2 of 2)

- Jumping (forward and back) is supported between activities in a generalized flow, sequence, or single thread or branch of a parallel flow
 - Source of a jump is a basic activity
 - Target of a jump can be a basic or structured activity
 - Jumping into nested constructs is not supported
- You can skip active or future basic activities in a process
 - Can immediately skip active activities (in a non-terminal state: running or ready)
 - Future activities can be marked to be skipped
 - Can skip basic activities only
 - Cannot skip structured activities (such as scope or sequence)
 - Can combine skip with jump
- If the activity is not reached, skipping an activity can be undone
- Specific business users and administrators can be authorized to initiate jumps, or to skip certain activities in a scope or collaboration scope
- You can update process variables at run time
 - Can use BPEL Explorer or Business Space to set variables for running instance
 - Useful in repair scenarios where inconsistent data must be changed

Business process handlers, runtime behavior, and clients

© Copyright IBM Corporation 2018

Runtime process dynamicity (2 of 2)

For a complete list of repairable items in business processes, see the product documentation.

Installing and uninstalling business process applications

- In the IBM Integration Designer test environment:
 - Server runs in “development mode” by default
 - Applications are installed and uninstalled easily by using add or remove projects
 - Removing applications does not require terminating and deleting instances, or stopping process templates
- On a production server:
 - Not necessary to stop a process template or task template before uninstalling an application; the template is stopped automatically
 - If uninstallation fails, the template is restarted
 - If instances are running, uninstallation fails
 - A check is done after the template is stopped
 - Applications can be removed by using a Jacl script even if instances exist in various states (it cleans up templates and instances, but should be used only during testing)

General Properties	
Name	server1
Node Name	widNode
<input checked="" type="checkbox"/> Run in development mode	

Installing and uninstalling business process applications

Business processes, as deployable artifacts, are contained within an EAR file and exposed as SCA components. They are installed in the normal application installation process. No more steps are necessary when installing an application that includes a business process. If you uninstall an application that contains a business process, you must first stop all running instances of that business process. This check is done to prevent the loss of state from a long-running business process by accidentally removing the application. Uninstalling a business process is a three-step process:

1. Ensure that all instances are completed and are removed.
2. Stop the template that represents that business process. (It is done automatically when the administrative console is used to uninstall the application.)
3. Uninstall the application.

This process is made easier in the development environment by using the “Run in development mode” option. By using this option, you can uninstall and update the application without having to terminate all business process instances. You can enable this feature for testing in the IBM Integration Designer environment, but it is not necessary to enable it in a production environment.

By using the `bpcTemplate.jacl` script, you can use the `-force` flag to uninstall applications that comprise processes or tasks with running instances. It is not meant for production environments except as a last resort. Removing applications along with running instances must be done during testing only.

Deletion of completed instances

- Deletion of completed instances as specified in process or task properties
 - **Yes:** Delete an instance when in a FINISHED, TERMINATED, or FAILED state
 - **On successful completion** (default): Delete only if not in FAILED state
 - **Completed after:** Delete instances that are complete after the specified date or time
 - **No:** Do not delete
- Cleanup service allows scheduled deletion of instances and tasks
 - Specify administratively which instances should be deleted and when
- Cleanup service configuration specifies:
 - When and how long (in minutes) the cleanup service should run
 - How many instances to delete in one transaction
 - Which instances should be deleted
- Cleanup job configuration specifies:
 - Instances that are going to be deleted
 - Time instances are kept after completion before deletion by the cleanup service
 - Jobs that run in the order that is listed

Deletion of completed instances

The following are many ways to delete process instances:

- Deletion of completed instances can be specified when defining business processes in IBM Integration Designer.
- Completed process instances can be deleted administratively either by using the `deleteCompletedProcessInstances` script, or by using the corresponding MBean interface with the `wsadmin` command-line tool.
- Administrators can use the BPEL Process Choreographer Explorer to delete selected instances (not ideal for handling large numbers of instances).
- Business Process Choreographer APIs (EJB, JMS, web services, REST) can be used to delete instances.
- You can use the cleanup service to delete instances.

Options for automatic deletion in IBM Integration Designer include:

- Do not automatically delete the process instance upon completion.
- Delete the process instance upon its completion (state: FINISHED, FAILED, or TERMINATED).
- Delete the process instance only if it completed successfully (state: FINISHED).

Unit summary

- List and describe the available handlers and error-processing activities
- Describe the runtime behavior of business processes
- Describe the administrative options and types of client access that are available for business processes

Checkpoint questions

1. What is a business process template?
2. What client would you use to view information about the state of running processes?
3. List three differences between the runtime behavior of a long-running process and a microflow.
4. What is the purpose of the cleanup service?
5. What is the purpose of the fault handler?
6. What is the purpose of a compensation handler?

Checkpoint questions

Checkpoint answers

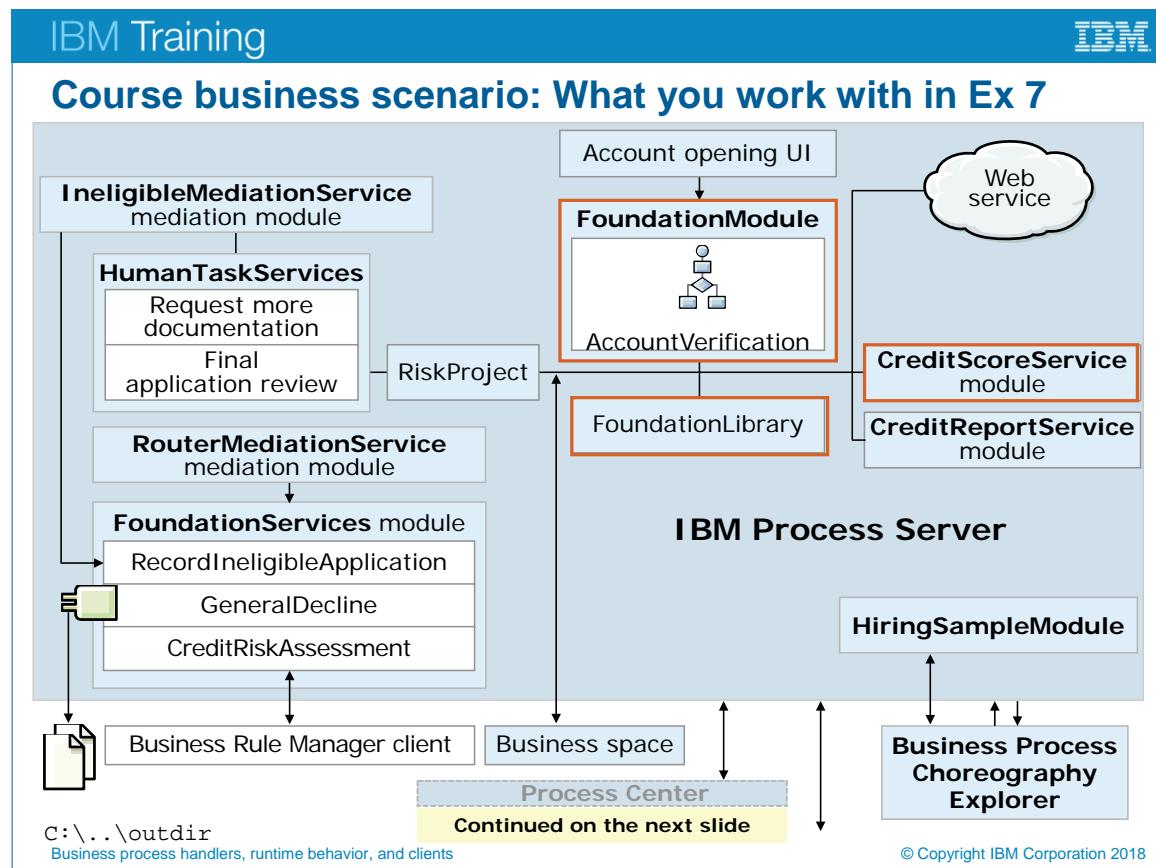
1. Process templates describe the process model. The Business Flow Manager uses the template to create instances of the process.
2. BPEL Process Choreographer Explorer or Business Space.
3. Microflows do not persist. No crash recovery is available for microflows. Microflows do not support parallelism. Microflows cannot be interrupted, are synchronous, and do not support correlation.
4. It allows the scheduled deletion of process instances and tasks administratively so you do not have to specify deletion behavior in the application.
5. A fault handler is a collection of specific activities that run when a fault (expected error) is thrown on the activity or scope. Fault handlers use catch and catch all elements to process error conditions in a process.
6. If an unexpected problem occurs, compensation handlers provide rollback capability for long-running processes and consist of all the steps that must be taken to restore the process to a balanced state.

Checkpoint answers

Exercise 7: Creating a business process, part III

After completing this exercise, you should be able to:

- Use data maps to transform process data
- Use context variables to create a runtime process description
- Assemble an SCA application that contains a business process
- Test a business process in the IBM Integration Designer test environment



Course business scenario: What you work with in Ex 7

Components that are required for Exercise 7

Prebuilt components that are imported in the lab:

- 1. FoundationModule**
- 2. CreditScoreService**
- 3. FoundationLibrary**
- 4. AccountVerification**

- BPEL process that you built in Exercises 6 and 7

New components that you create in the lab:

- 1. AccountVerification**

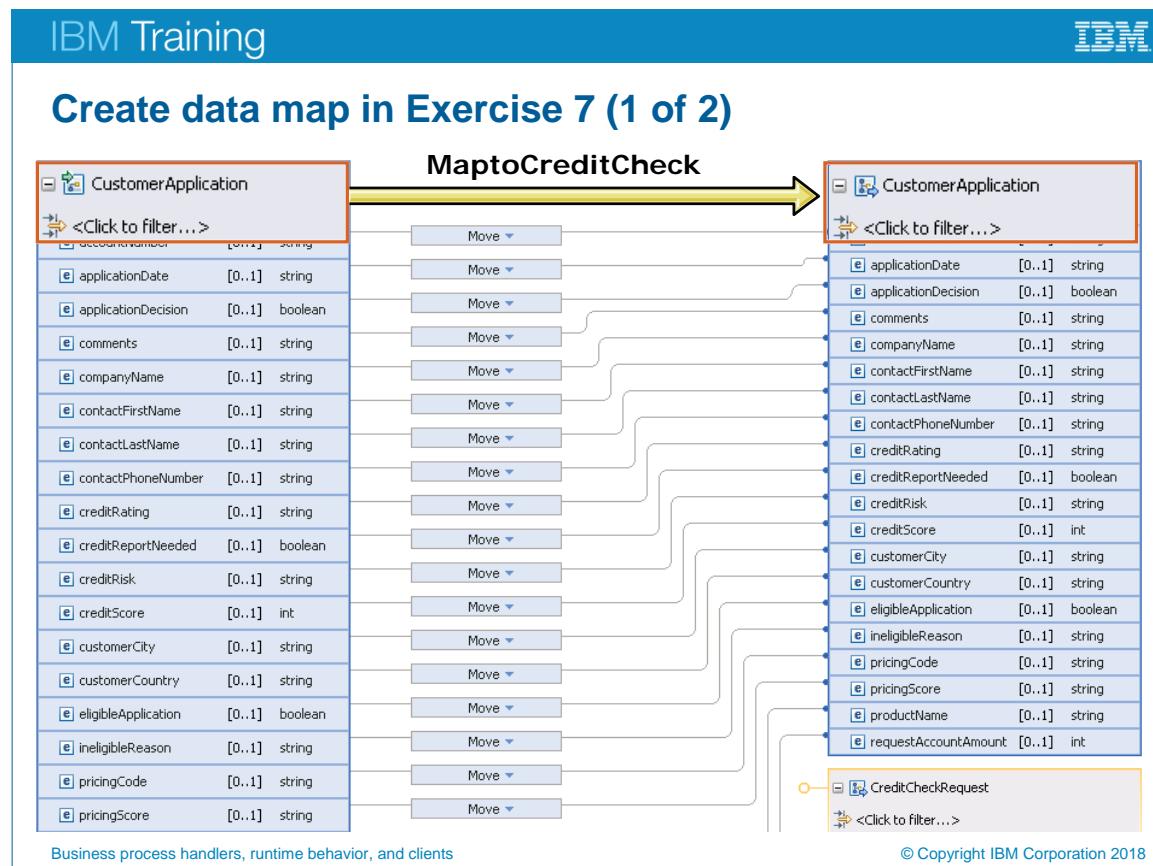
- BPEL process that you completed in Exercise 7
- You complete implementation of the process in this lab

Note:

The next slide lists all of the subcomponents of the **AccountVerification** process that are built in this lab.

Components that are required for Exercise 7

In this exercise, you create data maps to transform business objects between service requesters and providers. You also complete the core business logic for several of the services that the **AccountVerification** business process invokes. After you complete the service logic, you assemble the application that contains the **AccountVerification** process, and you test it.



Create data map in Exercise 7 (1 of 2)

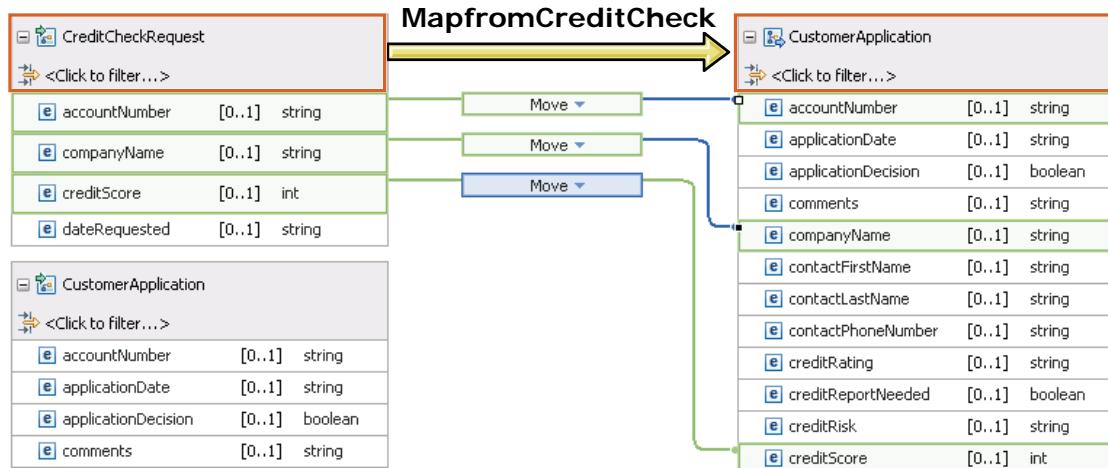
The AccountVerification business process that you are implementing receives and operates on a CustomerApplication business object input. During process execution, your business process calls an external web service, CreditScoreService, which provides the credit score that is needed to determine the customer's credit risk. Unlike the AccountVerification process (which uses a CustomerApplication business object input), the CreditScoreService interface uses a CreditCheckRequest business object for both the input and the output.

Because of this disparity, before you can invoke the CreditScoreService, you must transform the CustomerApplication business object into a CreditCheckRequest business object input. When the credit score is returned, you must transform the output from a CreditCheckRequest business object into a CustomerApplication business object. Transforming the data before and after service invocations in a business process can be done easily by using the data map activity in IBM Integration Designer.

The MaptoCreditCheck data map moves data from the CustomerApplicationVariable input to both a CustomerApplicationVariable output and a CreditCheckVariable output.

You transform the data into both objects so that you can preserve the existing data in CustomerApplicationVariable and merge the response data back into it.

Create data map in Exercise 7 (2 of 2)



Business process handlers, runtime behavior, and clients

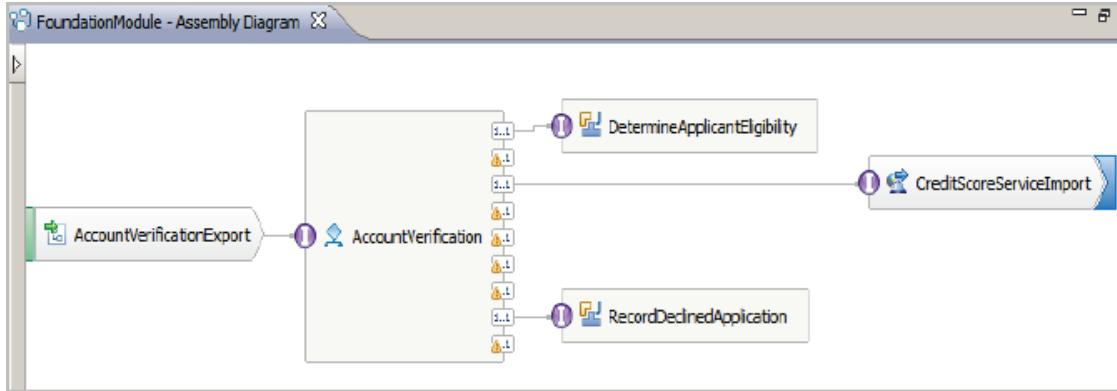
© Copyright IBM Corporation 2018

Create data map in Exercise 7 (2 of 2)

You create a **MapfromCreditCheck** data map that merges data from the **CreditCheckVariable** and **CustomerApplicationVariable** inputs into a **CustomerApplicationVariable** output.

IBM Training 

Assemble the SCA application that contains the business process in Exercise 7



The diagram illustrates the assembly of SCA components within a FoundationModule. It features a central composite component labeled 'AccountVerification' with several outgoing ports. One port connects to a 'DetermineApplicantEligibility' service. Another port connects to a 'CreditScoreServiceImport' service. A third port connects to a 'RecordDeclinedApplication' service. An external 'AccountVerificationExport' component is shown with an incoming port that connects to the 'AccountVerification' component.

Business process handlers, runtime behavior, and clients © Copyright IBM Corporation 2018

Assemble the SCA application that contains the business process in Exercise 7

In the assembly diagram, you wire the SCA components into an integrated application that is deployed to the runtime environment.

IBM Training IBM

Test the created application by deploying to a Process Server runtime

The screenshot shows the 'Deployment Location' dialog box and the 'Servers' view in IBM Studio.

Deployment Location Dialog:

- Select a Deployment Location:** Specify a runtime location where this test will deploy.
- Deployment location:** IBM Process Servers (selected) and IBM Process Server v8.5.6 at localhost.
- Mode:** Run
- Checkboxes:** Use this location as the default and do not ask again (checked).
- Note:** To change the default location elsewhere, open the Properties window for the project and select Business Integration > Integration Test Client.

Servers View:

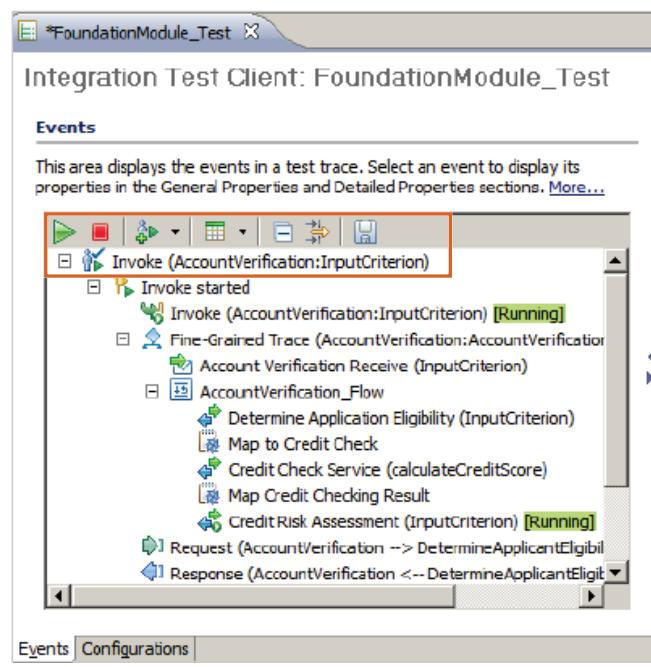
- Properties, Problems, Server Logs, Servers tabs.
- Server list:
 - IBM Process Server v8.5.6 at localhost [Started, Synchronized]
 - CreditScoreServiceApp [Started, Synchronized]
 - FoundationModuleApp [Stopped, Synchronized]
- Buttons: Cancel, Start, Stop, Refresh, etc.

Business process handlers, runtime behavior, and clients © Copyright IBM Corporation 2018

Test the created application by deploying to a Process Server runtime

Use the test client in Exercise 7

- Use the Integration Test client to run and test the **AccountVerification** process



Business process handlers, runtime behavior, and clients

© Copyright IBM Corporation 2018

Use the test client in Exercise 7

Exercise 7: Creating a business process, part III

Purpose:

In this exercise, you use IBM Integration Designer to finish building a complex BPEL business process. You use data maps to transform data, and you implement services that the process invokes. After completing the process, you add the business process to an SCA assembly diagram and test it.

The Web Services Business Process Execution Language (WS-BPEL) provides syntax for specifying the behavior of a business process in a platform-independent manner. It is used to coordinate a series of service invocations to fulfill a business task. Although the language provides conditional and control logic, most of the work is done through the invoked services.

Combining BPEL with the SCA programming model allows for the coordination of SCA services into much larger units of work. Individual SCA services can be brought together and can benefit from the advanced capabilities of WS-BPEL.

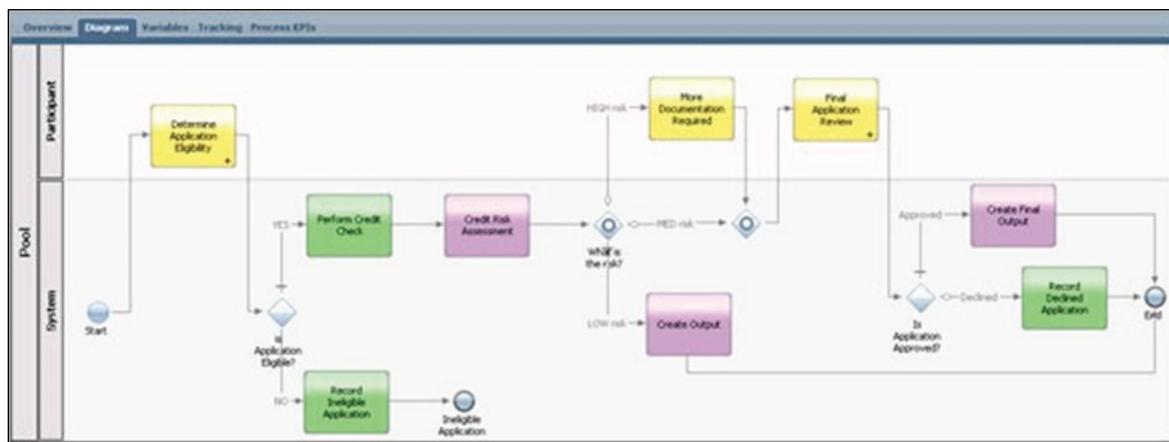
Requirements

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

In this exercise, you create data maps to transform business objects between service requesters and providers. You also complete the core business logic for several of the services that the `AccountVerification` business process invokes. When you complete the service logic, you assemble the application that contains the `AccountVerification` process, and you test it.

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: Use data maps to transform process data

The AccountVerification business process that you are implementing receives and operates on a CustomerApplication business object input. During process execution, your business process calls an external web service, CreditScoreService, which provides the credit score that is needed to determine the customer's credit risk. Unlike the AccountVerification process (which uses a CustomerApplication business object input), the CreditScoreService interface uses a CreditCheckRequest business object for both the input and the output.

Because of this disparity, before you can invoke the CreditScoreService, you must transform the CustomerApplication business object into a CreditCheckRequest business object input. When the credit score is returned, you must transform the output from a CreditCheckRequest business object into a CustomerApplication business object.

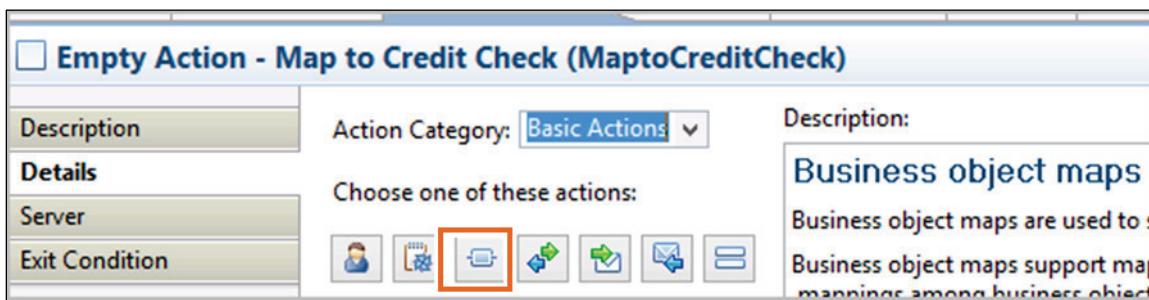
The data map activity can be used to easily transform the data before and after service invocations in a business process. For more information about working with data maps, see the product documentation.

1. Create data maps to transform inputs and outputs.

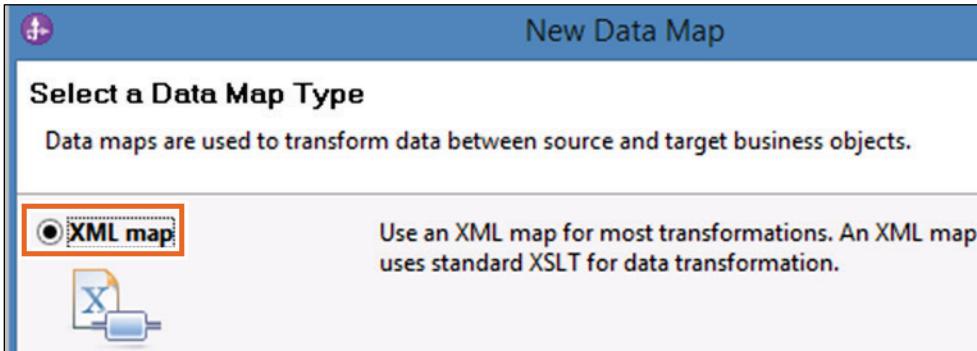
1. On your desktop, open the **Exercise Shortcuts** folder.
2. Double-click the **Exercise 7** shortcut.

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. Open the AccountVerification business process.
 1. In the Business Integration view, expand FoundationModule > Integration Logic > BPEL Processes > AccountVerification.
 2. Double-click **AccountVerification** to open the process editor.
3. Change the Name of the Map to Credit Check empty action to: **MaptoCreditCheck**
 1. Select the **Map to Credit Check** empty action and switch to the **Description** tab in the **Properties** view.
 2. Change the value in the **Name** field from **EmptyAction** to: **MaptoCreditCheck**
 3. Save your changes.
You will change the implementation of the Map to Credit Check empty action to a data map. Create a **MaptoCreditCheck** data map that moves data from the **CustomerApplicationVariable** input to both a **CustomerApplicationVariable** output and a **CreditCheckVariable** output.
4. Transform the data into both objects so you can preserve the existing data in **CustomerApplicationVariable** and merge the response data back into it.
 1. Ensure that the **Map to Credit Check** empty action is selected, and switch to the **Details** tab in the **Properties** view.
 2. For **Choose one of these actions**, click the **Data Map** icon.



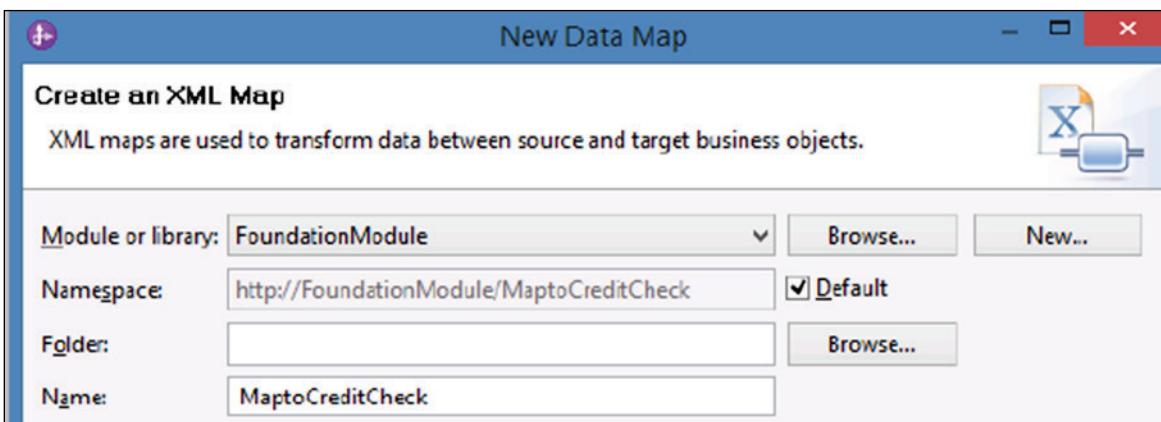
3. At the “Select a Data Map Type” dialog box, select **XML map**.



4. Click **Next**.

5. At the “Create an XML Map” dialog box, follow these instructions:

- Verify that **Module or Library** is set to: FoundationModule
- Leave the **Folder** field empty
- In the **Name** field, type: MptoCreditCheck

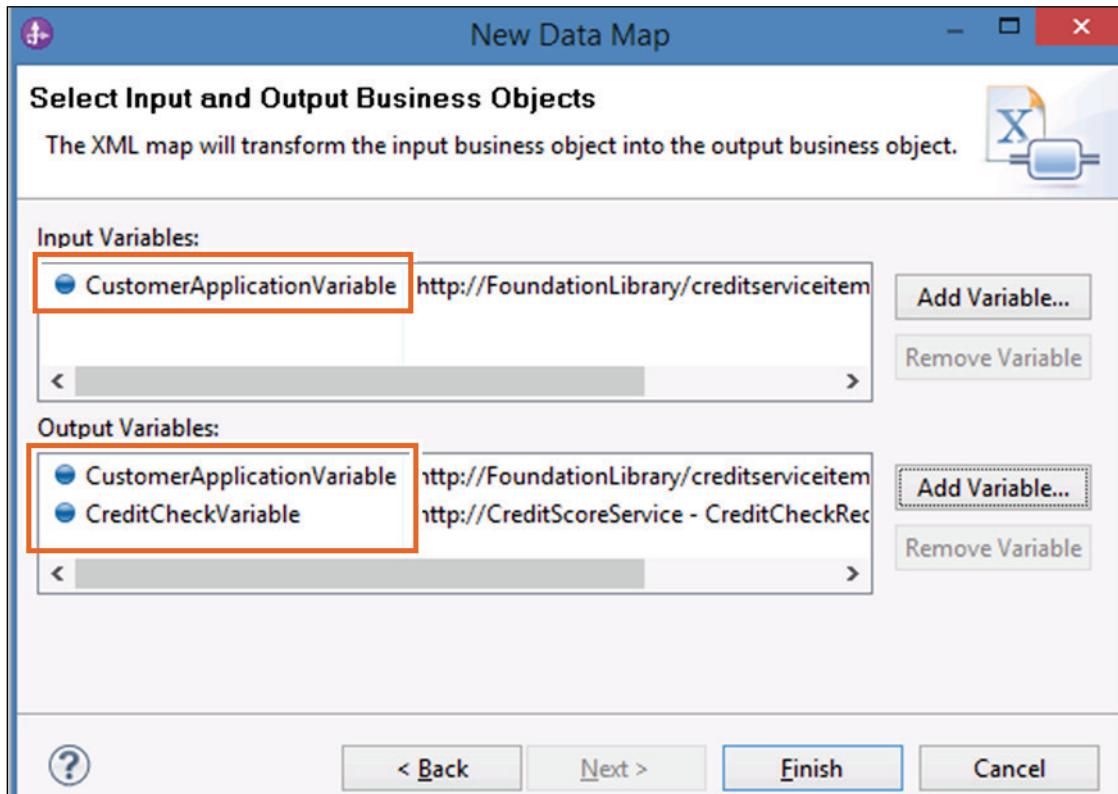


6. Click **Next**.

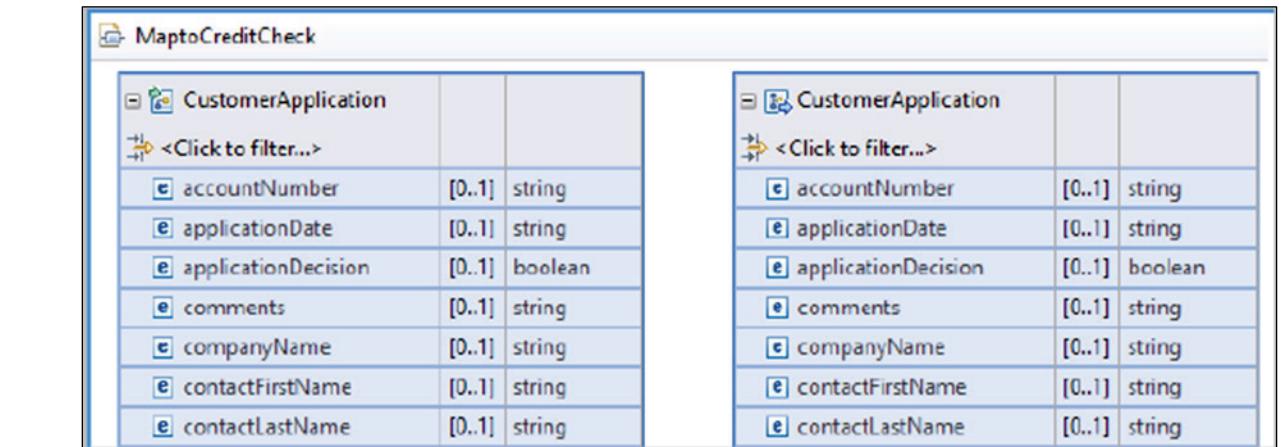
7. At the “Select Input and Output Business Objects” dialog box, do the following steps:

- For **Input Variables**, click **Add Variable** and select **CustomerApplicationVariable**
- In the “Select a Variable” dialog box, click **OK**.
- For **Output Variables**, click **Add Variable** and select **CustomerApplicationVariable** in the “Select a Variable” dialog box. Click **OK**.
- For **Output Variables**, click **Add Variable** again and select **CreditCheckVariable** in the “Select a Variable” dialog box. Click **OK**.

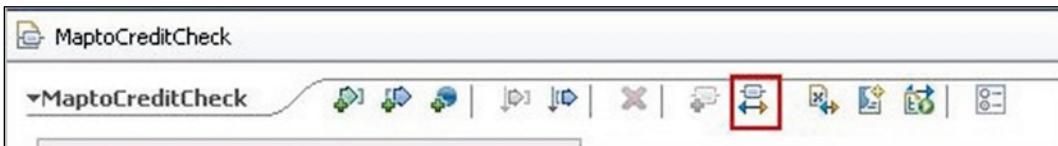
You cannot select more than one variable at the same time.



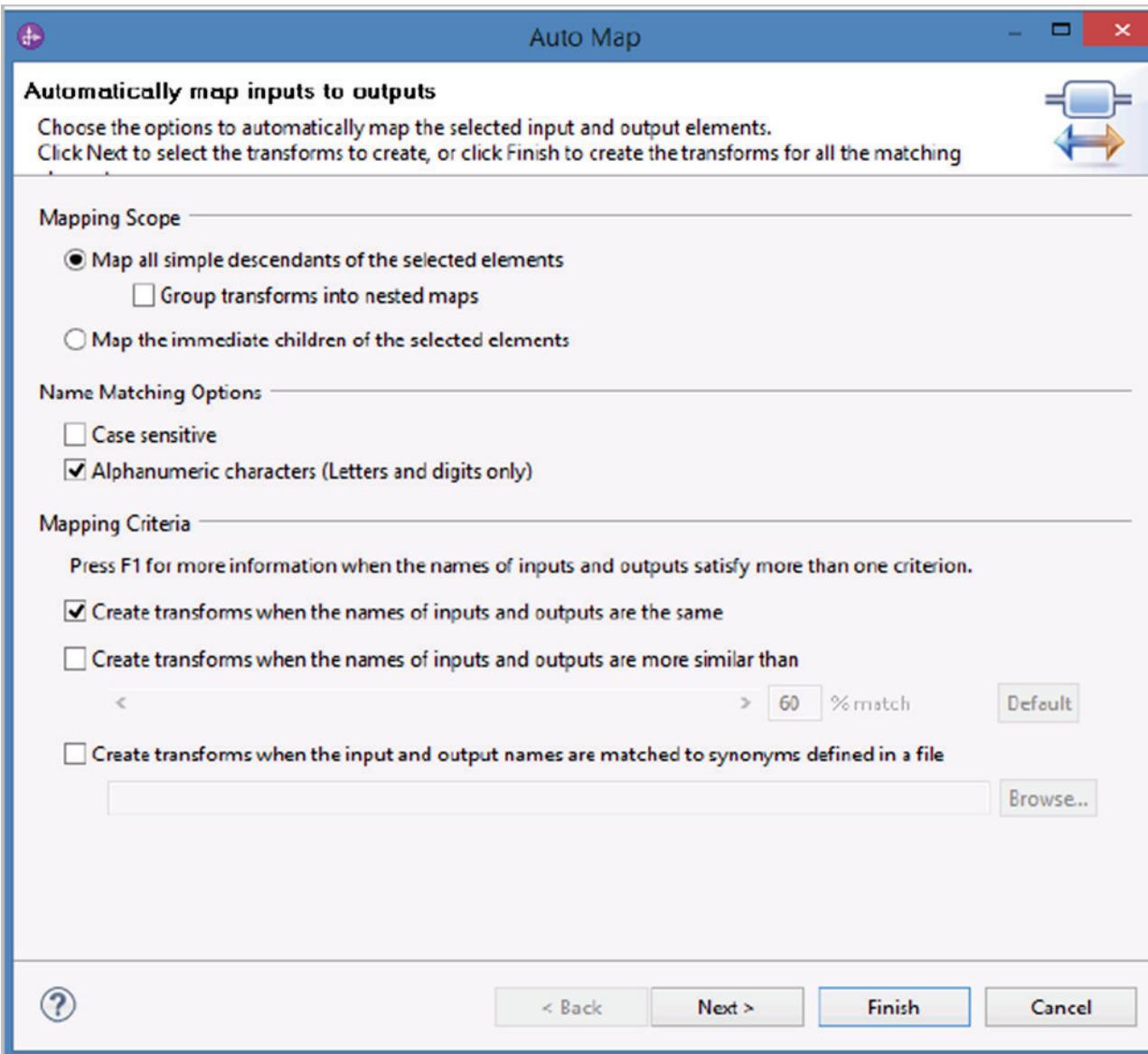
8. Click **Finish** to open the XML mapping editor.
5. Use Move transformations to move the data from the **CustomerApplication** input variable to the **CustomerApplication** output variable.
You use the **Automap input to output** feature to create the transformations automatically.



2. Click the Automap input to output icon.

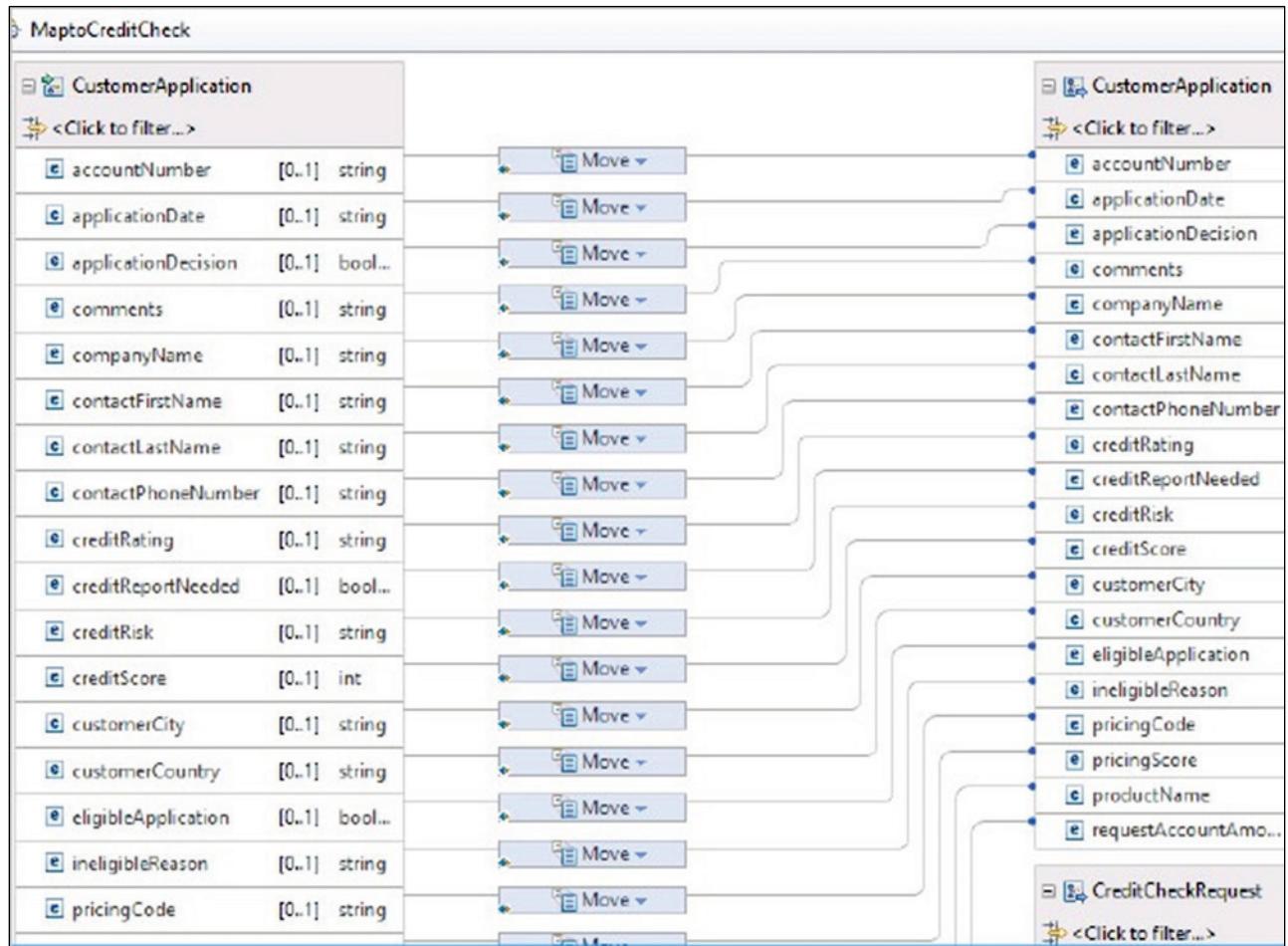


3. In the Auto Map window, accept defaults and click Finish.



This action uses **Move** transformations to map each of the fields from the CustomerApplication input to the CustomerApplication output.

Your map resembles the following diagram. The automap automatically mapped the accountNumber, companyName, and creditScore fields to both output variables. Some transformations are not visible in the graphic because of space constraints.

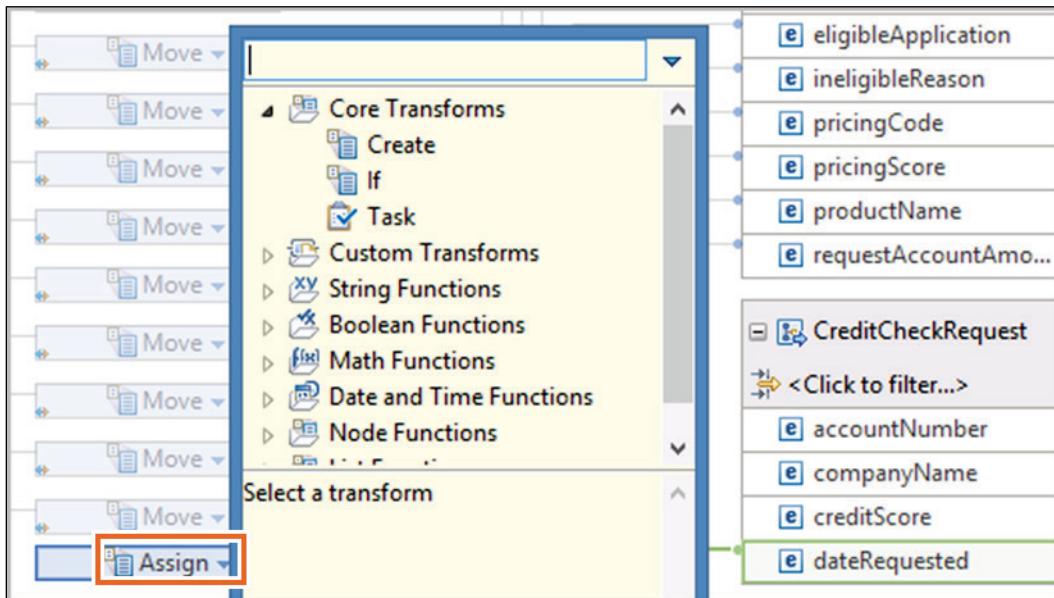


Note: You can cut, copy, and paste transformations in the mapping editor. This option speeds development when reusing maps. For example, when building new maps from old maps, you can copy the schema from the old map to the new map or submap. If the target schema does not match the source schema that you copied, you are not allowed to paste.

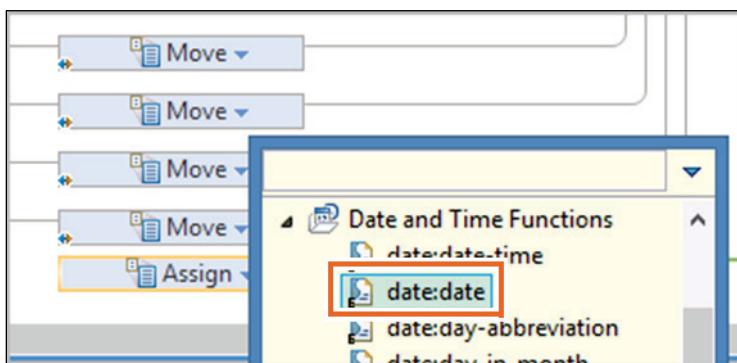
6. Create an Assign transformation that places the current date in the dateRequested attribute of the CreditCheckRequest output.

 1. Right-click **dateRequested** in the CreditCheckRequest output and click **Create Assign** from the menu.

2. Click the down arrow in the Assign transformation to open the dialog box.



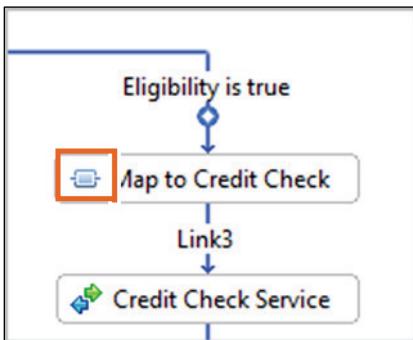
3. Expand Date and Time Functions and select date:date.



Your completed map resembles the following diagram. All transformations are not visible in the graphic because of space constraints.



7. Use Move transformations to manually map the accountNumber, companyName, and creditScore fields from the CustomerApplication input to the CreditCheckRequest output.
1. Hover over the **creditScore** field in the **CustomerApplication** input (on the left side of the diagram).
 2. Drag the orange, circular handle from **creditScore** in the **CustomerApplication** input to **creditScore** in the **CreditCheckRequest** output. This action creates a **Move** transformation.
 3. In the **CustomerApplication** input, right-click **accountNumber** and click **Create Connection**.
 4. Click **accountNumber** in the **CreditCheckRequest** output. This action creates a **Move** transformation.
 5. In the **CustomerApplication** input, right-click **companyName** and click **Create Connection**.
 6. Click **companyName** in the **CreditCheckRequest** output. This action creates a **Move** transformation.
 7. Save and close the data map. Verify that the **Problems** view shows no errors. If you see any errors, then verify that you created the data map correctly.
 8. Save the changes to the business process. The icon for the Map to Credit Check activity changes to a data map icon.



8. Change the Name of the Map Credit Checking Result empty action to:

`MapCreditCheckingResult`

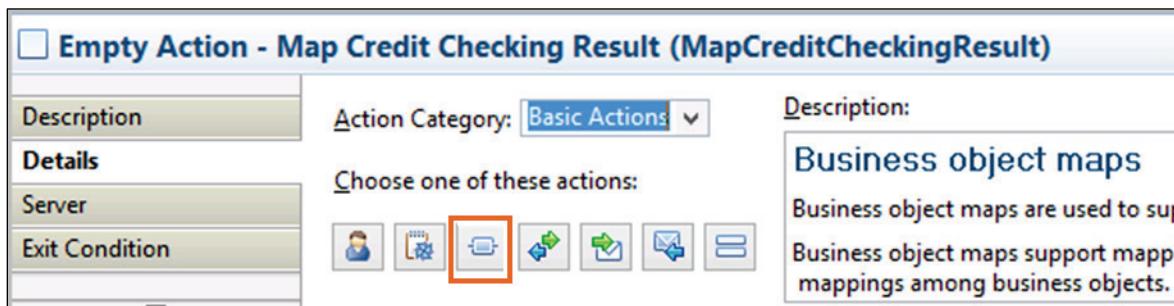
1. Select the **Map Credit Checking Result** empty action and switch to the **Description** tab in the **Properties** view.
2. Change the value in the **Name** field from `EmptyAction1` to:

MapCreditCheckingResult

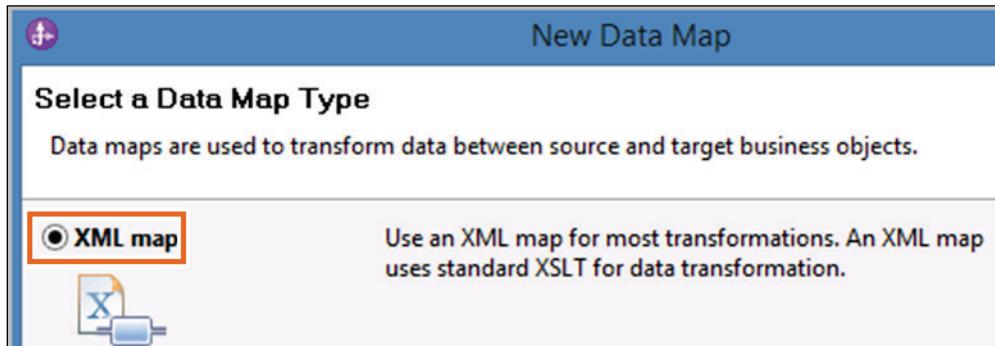
3. Save your changes.
9. Change the implementation of the Map Credit Checking Result empty action to a data map.

Create a MapfromCreditCheck data map that merges data from the CreditCheckVariable and CustomerApplicationVariable inputs into a CustomerApplicationVariable output.

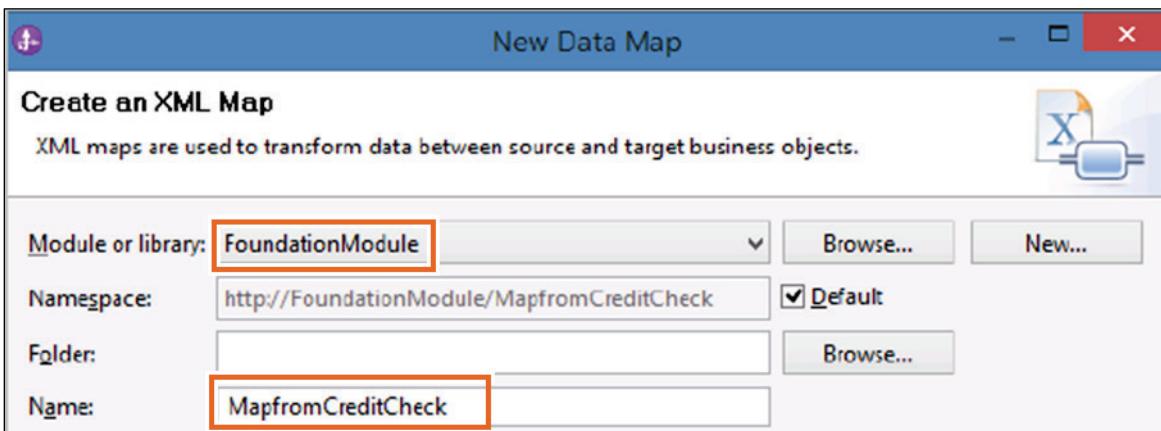
1. Select the **Map Credit Checking Result** empty action and switch to the **Details** tab in the **Properties** view.
2. For **Choose one of these actions**, click the **Data Map** icon.



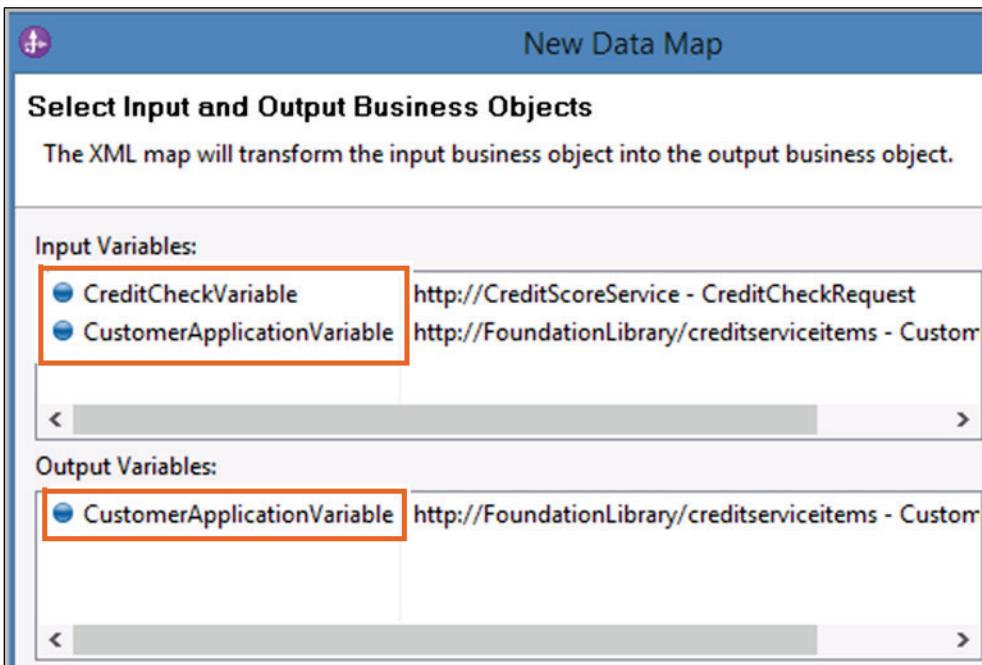
3. At the “Select a Data Map Type” dialog box, select **XML map**.



4. Click **Next**.
5. At the “Create an XML Map” dialog box, enter the following information.
 - Verify that **Module or Library** is set to: FoundationModule
 - Leave the **Folder** field empty
 - In the **Name** field, type: MapfromCreditCheck



6. Click **Next**.
7. At the “Select Input and Output Business Objects” dialog box, do these steps.
 - For **Input Variables**, click **Add Variable** and select **CreditCheckVariable** in the “Select a Variable” dialog box.
 - For **Input Variables**, click **Add Variable** and select **CustomerApplicationVariable** in the “Select a Variable” dialog box.
 - For **Output Variables**, click **Add Variable** and select **CustomerApplicationVariable**.



8. Click **Finish** to open the data map editor.

10. Map the Credit Check result data to the CustomerApplicationVariable.

1. Hold down the Control key and select the **CustomerApplication** and **CreditCheckRequest** input variables and the **CustomerApplication** output variable.
2. Click the **Automap input to output** icon.

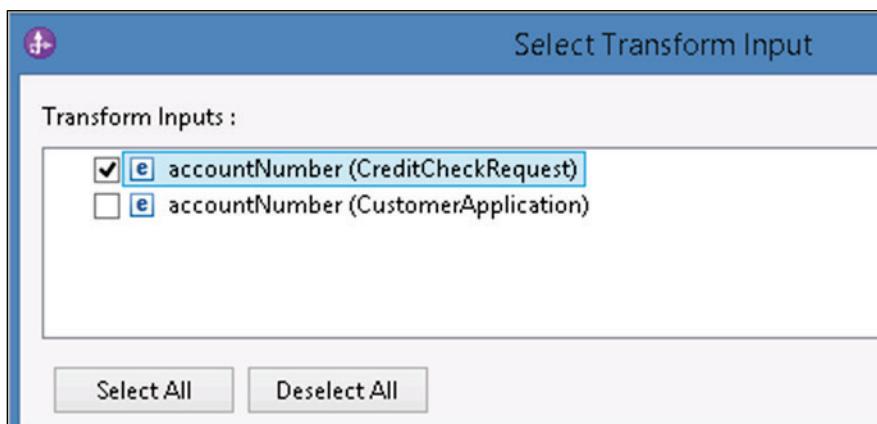
The screenshot shows the 'MapfromCreditCheck' interface. At the top, there are two tables: 'CreditCheckRe...' on the left and 'CustomerApplication' on the right. Both tables have a 'Click to filter...' button. Below these are two larger tables: 'CustomerAppl...' on the left and 'CustomerApplication' on the right. The 'CustomerAppl...' table has a 'Click to filter...' button. The 'CustomerApplication' table on the right has a 'Click to filter...' button. In the center, there is a large grid area where mappings can be defined between the variables in the two tables. A red box highlights the 'Automap input to output' icon located at the top right of the interface.

MapfromCreditCheck		
CreditCheckRe...		
<Click to filter...>		
accountNumber	[0..1]	string
companyName	[0..1]	string
creditScore	[0..1]	int
dateRequested	[0..1]	string
CustomerAppl...		
<Click to filter...>		
accountNumber	[0..1]	string
applicationDate	[0..1]	string
applicationDec...	[0..1]	bool...
comments	[0..1]	string
companyName	[0..1]	string
contactFirstName	[0..1]	string
contactLastName	[0..1]	string
contactPhoneNumber	[0..1]	string
creditRating	[0..1]	string
creditReportN...	[0..1]	bool...
creditRisk	[0..1]	string
creditScore	[0..1]	int
customerCity	[0..1]	string
customerCoun...	[0..1]	string
CustomerApplication		
<Click to filter...>		
accountNumber	[0..1]	string
applicationDate	[0..1]	string
applicationDecision	[0..1]	boolean
comments	[0..1]	string
companyName	[0..1]	string
contactFirstName	[0..1]	string
contactLastName	[0..1]	string
contactPhoneNumber	[0..1]	string
creditRating	[0..1]	string
creditReportNeeded	[0..1]	boolean
creditRisk	[0..1]	string
creditScore	[0..1]	int
customerCity	[0..1]	string
customerCountry	[0..1]	string
eligibleApplication	[0..1]	boolean
ineligibleReason	[0..1]	string
pricingCode	[0..1]	string
pricingScore	[0..1]	string
productName	[0..1]	string
requestAccountAmo...	[0..1]	int

3. Accept the defaults in the Auto Map dialog box. Click **Next**.
4. Click **accountNumber** in the Transform Outputs list. Click **Edit**.

Transforms found:			
Transform Outputs	Transform Inputs	Input count	
CustomerApplication			Edit
<input checked="" type="checkbox"/> <input type="checkbox"/> accountNumber	accountNumber (CreditCheckRequest), accountN...	2	
<input checked="" type="checkbox"/> <input type="checkbox"/> applicationDate	applicationDate (CustomerApplication)	1	
<input checked="" type="checkbox"/> <input type="checkbox"/> applicationDecision	applicationDecision (CustomerApplication)	1	
<input checked="" type="checkbox"/> <input type="checkbox"/> comments	comments (CustomerApplication)	1	
<input checked="" type="checkbox"/> <input type="checkbox"/> companyName	companyName (CreditCheckRequest), companyN...	2	
<input checked="" type="checkbox"/> <input type="checkbox"/> contactFirstName	contactFirstName (CustomerApplication)	1	
<input checked="" type="checkbox"/> <input type="checkbox"/> contactLastName	contactLastName (CustomerApplication)	1	
<input checked="" type="checkbox"/> <input type="checkbox"/> contactPhoneNumber	contactPhoneNumber (CustomerApplication)	1	

5. Clear the check box next to **accountNumber (CustomerApplication)** and click **OK**.

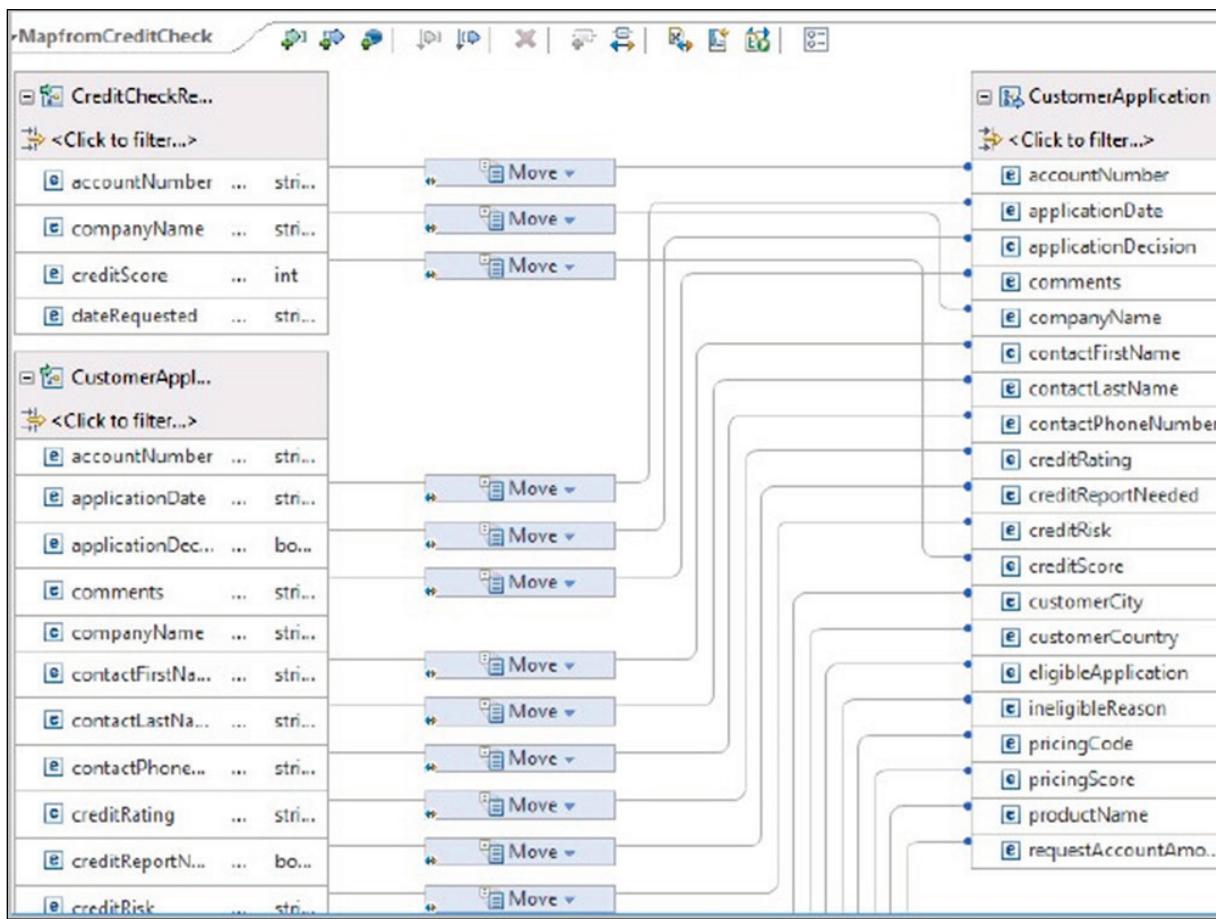


6. Click **companyName** in the Transform Outputs list. Click **Edit**.
 7. Clear the check box next to **companyName (CustomerApplication)**. Click **OK**.
 8. Click **creditScore** in the Transform Outputs list. Click **Edit**.
 9. Clear the check box next to **creditScore (CustomerApplication)**. Click **OK**.
 10. Verify the edits that you made. No input field has an Input count other than 1.

Auto Map		
Select transforms to create		
Transform Outputs	Transform Inputs	Input count
<input checked="" type="checkbox"/>  CustomerApplication		1
<input checked="" type="checkbox"/>  accountNumber	accountNumber (CreditCheckRequest)	1
<input checked="" type="checkbox"/>  applicationDate	applicationDate (CustomerApplication)	1
<input checked="" type="checkbox"/>  applicationDecision	applicationDecision (CustomerApplication)	1
<input checked="" type="checkbox"/>  comments	comments (CustomerApplication)	1
<input checked="" type="checkbox"/>  companyName	companyName (CreditCheckRequest)	1
<input checked="" type="checkbox"/>  contactFirstName	contactFirstName (CustomerApplication)	1
<input checked="" type="checkbox"/>  contactLastName	contactLastName (CustomerApplication)	1
<input checked="" type="checkbox"/>  contactPhoneNumber	contactPhoneNumber (CustomerApplication)	1
<input checked="" type="checkbox"/>  creditRating	creditRating (CustomerApplication)	1
<input checked="" type="checkbox"/>  creditReportNeeded	creditReportNeeded (CustomerApplication)	1
<input checked="" type="checkbox"/>  creditRisk	creditRisk (CustomerApplication)	1
<input checked="" type="checkbox"/>  creditScore	creditScore (CreditCheckRequest)	1
<input checked="" type="checkbox"/>  customerCity	customerCity (CustomerApplication)	1
<input checked="" type="checkbox"/>  customerCountry	customerCountry (CustomerApplication)	1
<input checked="" type="checkbox"/>  eligibleApplication	eligibleApplication (CustomerApplication)	1
<input checked="" type="checkbox"/>  ineligibleReason	ineligibleReason (CustomerApplication)	1
<input checked="" type="checkbox"/>  pricingCode	pricingCode (CustomerApplication)	1
<input checked="" type="checkbox"/>  pricingScore	pricingScore (CustomerApplication)	1

11. Click **Finish**.

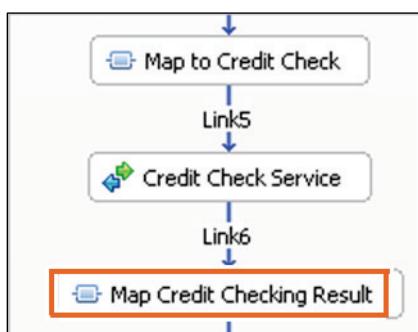
Your data map resembles the following figure. Some transformations are not visible in the graphic because of space constraints.



Note: You can also manually map inputs to outputs by clicking a field in the output list, then dragging a connector line to an input field. If you select an individual transformation (by clicking the transformation name or the connector), the connector and the mapped attributes are highlighted in green. You can use this display to verify the accuracy of the transformations that you created.

12. Save and close the data map.
13. Save the changes to the business process.

The icon for the **Map Credit Checking Result** activity changes to a data map icon.

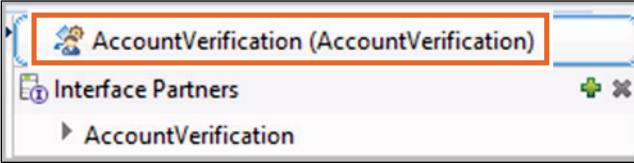
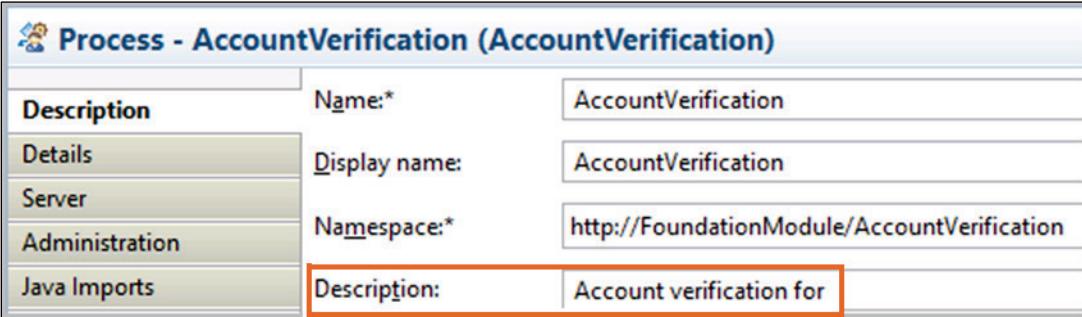


Task 2. Use context variables to create a runtime process description.

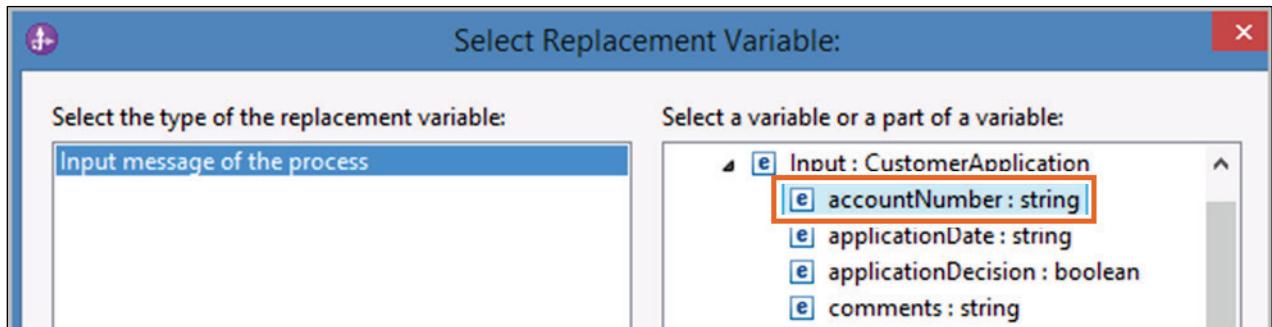
When used in descriptions for processes and human tasks, replacement expressions can be used to represent context variables that are fully resolved in the runtime environment. Context variables can come from many sources. For example, they can originate from input or output messages.

Context variables can be used in many places within the tool environment, including process descriptions. For more information about replacement expressions, see the product documentation.

In this portion of the exercise, you use context variables to create a process description. At run time, when the context variables are resolved, the replacement expression creates a meaningful description of process instances when you view them in clients such as the Business Process Choreographer Explorer. For process instances, you use information in the process input message to populate the process description at run time.

1. Create a process description that contains context variables.
 1. Click any blank space in the process editor *outside* the `AccountVerification_Flow` activity. You can also click the `AccountVerification` icon at the top of the tray.
 
 2. Switch to the **Description** tab in the **Properties** view.
 3. In the **Description** field, type the following text: Account verification for (add a space at the end of the text)
 
 4. To the right of the **Description** field, click **Insert Variable**.

5. In the “Select Replacement Variable” dialog box, in the “Select a variable or a part of a variable” window, expand **InputCriterionParameters : InputCriterion > Input : CustomerApplication**.
6. Select **accountNumber : string**.



Note the replacement variable syntax at the bottom of the dialog box.

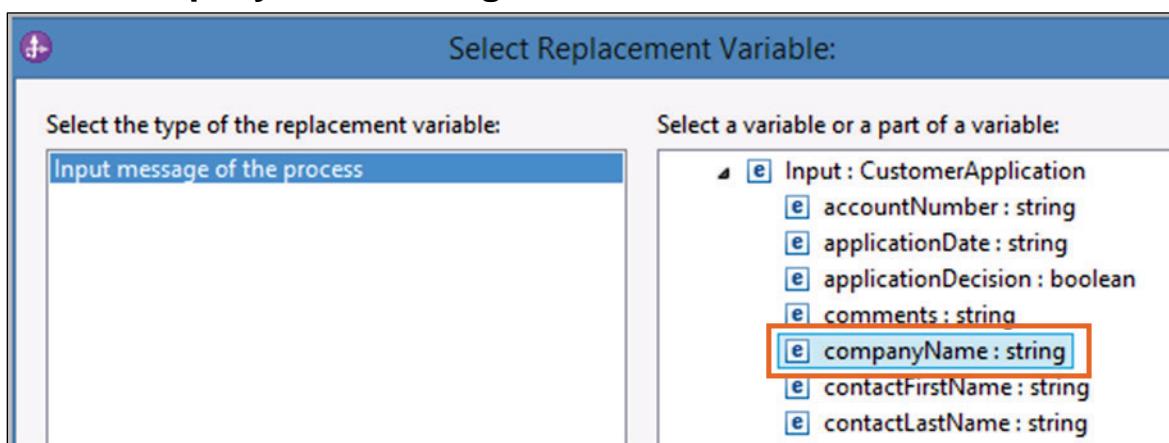
The replacement variable will be: %InputCriterionParameters\Input/accountNumber%

7. Click **OK**.

The replacement variable expression is displayed in the **Description** field.

Name:*	AccountVerification
Display name:	AccountVerification
Namespace:*	http://FoundationModule/AccountVerification
Description:	Account verification for%\\InputCriterionParameters\\Input/accountNumber%

8. Insert a space at the end of the **Description** field.
9. Click **Insert Variable**.
10. In the “Select Replacement Variable” dialog box, in the “Select a variable or a part of a variable” window, expand **InputCriterionParameters : InputCriterion > Input : CustomerApplication**.
11. Select **companyName : string**.



12. Click OK.

The completed expression resembles the following text:

```
Account verification for  
%InputCriterionParameters\Input/accountNumber%  
%InputCriterionParameters\Input/companyName%
```

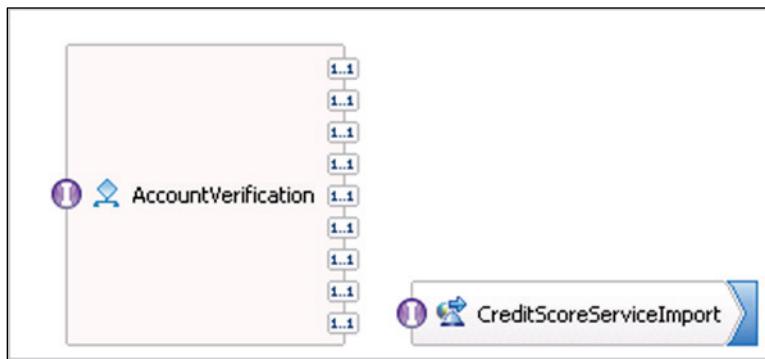
When you view a process instance at run time, this expression is resolved into a description such as: Account verification for IBM007 IBM

13. Save your changes.

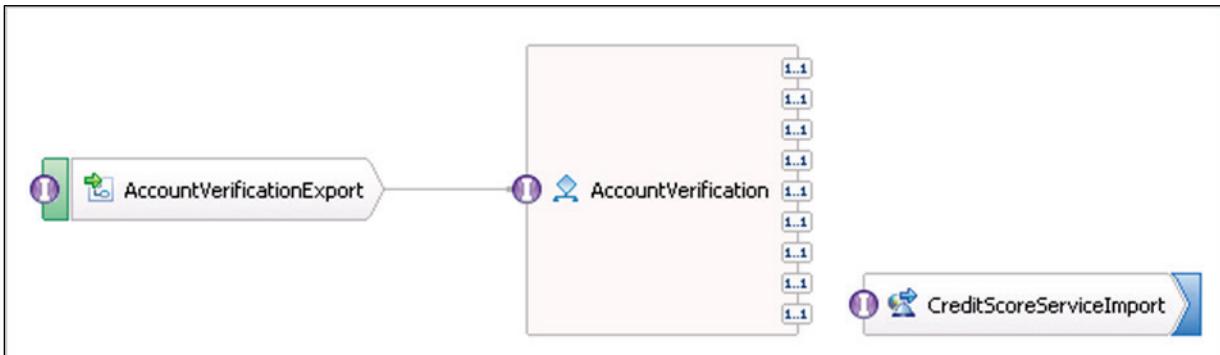
Task 3. Assemble an SCA application that contains a business process.

In this portion of the exercise, you modify the assembly diagram for **FoundationModule** and implement the services that activities in the business process invoke. Some of the services are implemented with simple Java code that is designed to simulate actual services. Others, such as human tasks and adapters, are implemented in later exercises.

1. Add the **AccountVerification** service component to the **FoundationModule** assembly diagram and generate an export component with an SCA binding.
 1. In the Business Integration view, expand **FoundationModule**.
 2. Double-click **Assembly Diagram** to open the assembly editor.
 3. Expand **FoundationModule > Integration Logic > BPEL Processes > AccountVerification**.
 4. Drag the **AccountVerification** business process onto the assembly diagram.



5. Right-click **AccountVerification** and click **Generate Export > SCA Binding** from the menu.



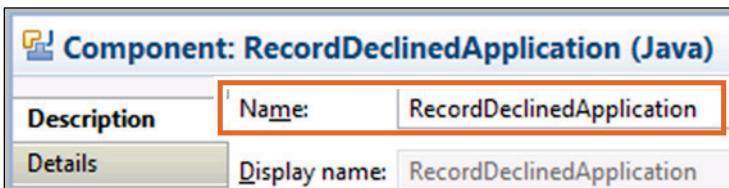
6. Save your changes.

Information: The references on the AccountVerification component are not wired to endpoints. Therefore, you see warnings in the Problems view. Ignore these warnings.

2. Add a Java component to the assembly diagram and name the component:

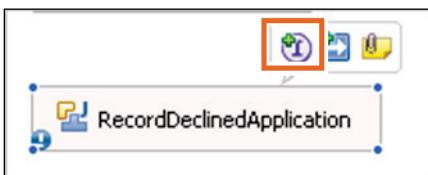
RecordDeclinedApplication

1. In the assembly editor palette, expand **Components** and select **Java**.
2. Click any blank space on the assembly diagram to add the component.
3. With the component that is selected, switch to the **Description** tab in the **Properties** view.
4. Change the **Name** to: RecordDeclinedApplication

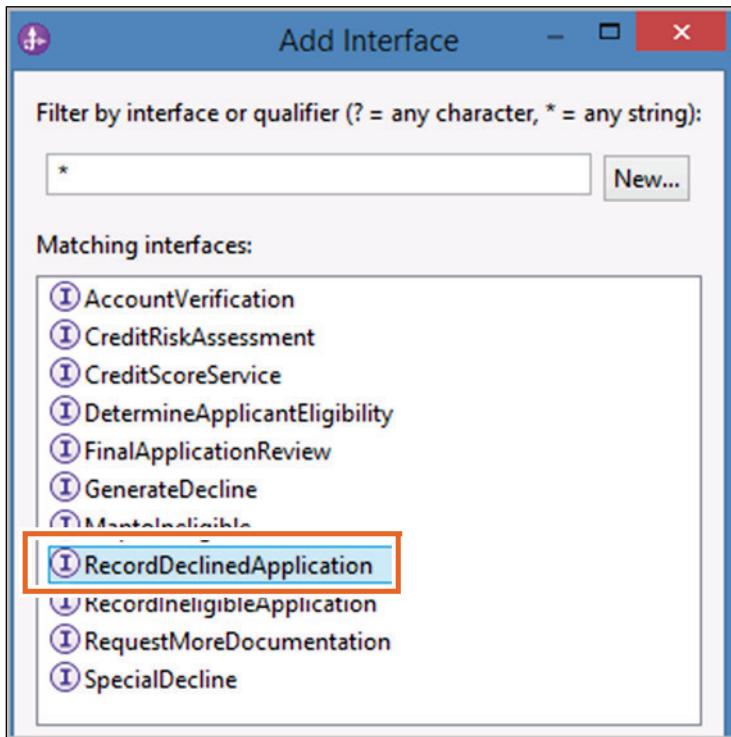


3. Add the RecordDeclinedApplication interface to the Java component.

1. Select the **RecordDeclinedApplication** component in the assembly diagram.
2. Click the **Add Interface** icon in the action bar.

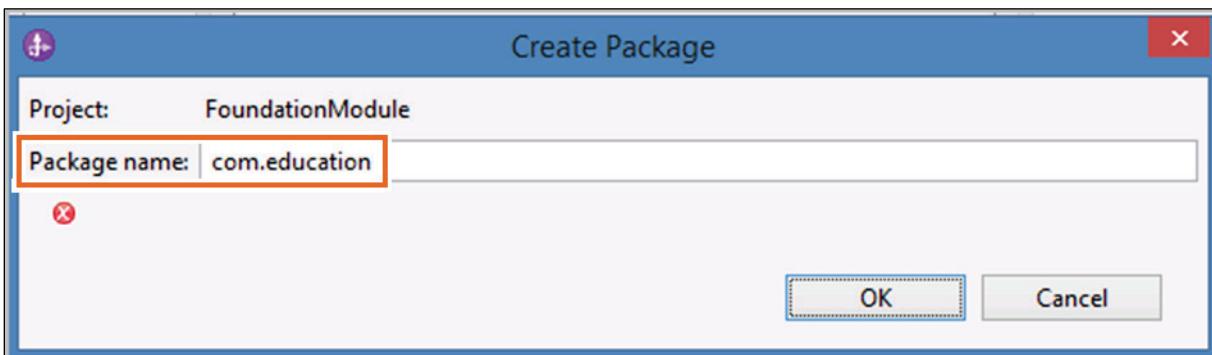


3. In the Add Interface dialog box, select **RecordDeclinedApplication**.



Information: By using the **Add Interface** dialog box, you can filter interfaces by type: Java, WSDL, or both.

4. Click **OK**.
4. Use the snippet in `c:\labfiles\Support Files\Ex7\RecordDeclinedApplicationImpl_snippet.txt` to generate the implementation for the Java component. Generate the implementation in a new package that is named: `RecordDeclinedApplication`
 1. Right-click the **RecordDeclinedApplication** Java component and click **Generate Implementation** from the menu.
 2. In the Generate Implementation dialog box, click **New Package**.
 3. In the Create Package dialog box, in the **Package Name** field, type: `com.education`



4. Click **OK**.
5. With **com.education** selected in the Generate Implementation dialog box, click **OK**.
This action opens `RecordDeclinedApplicationImpl.java` in the Java editor.
6. In the **RecordDeclinedApplicationImpl.java** tab, scroll to the `public void InputCriterion` declaration at the end of the file.
7. In Windows Explorer, browse to `C:\labfiles\Support Files\Ex7`.
8. Open `RecordDeclinedApplicationImpl_snippet.txt` in a text editor such as Notepad.
9. Copy the text in `RecordDeclinedApplicationImpl_snippet.txt` and paste the content over the comment lines (the green text that begins with `//`) in the `InputCriterion` method.

```
public void InputCriterion(DataObject input) {
    System.out.println("[Java] Record Declined Application - begins");
    String ret = "Compliance Logging: Application from customer "
        + input.getString("companyName") + " has been declined.";
    System.out.println("[Java] Record Declined Application - ends");
}
```

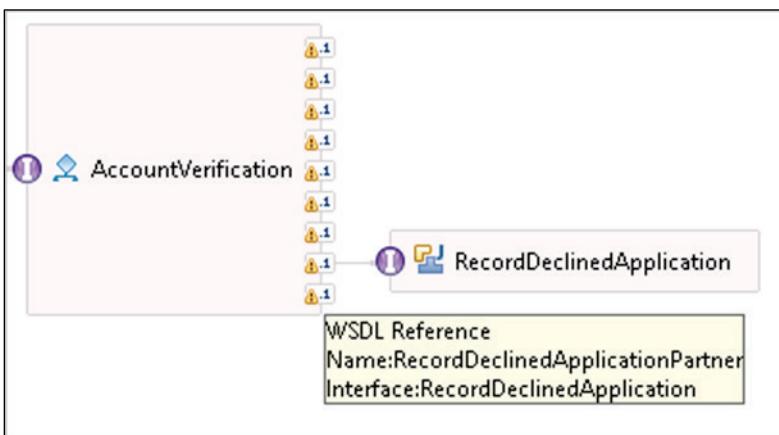
10. Alternatively, enter the following code:

```
System.out.println("[Java] Record Declined Application - begins");
String ret = "Compliance Logging: Application from customer " +
    input.getString("companyName") + " has been declined.";
System.out.println("[Java] Record Declined Application - ends");
```

11. Press **Ctrl+S** to save your changes and **Ctrl + W** to close the Java editor.
12. Close the `RecordDeclinedApplicationImpl_snippet.txt` file. Leave Windows Explorer open.

5. Wire the RecordDeclinedApplication Java component to the RecordDeclinedApplicationPartner reference of the AccountVerification process.

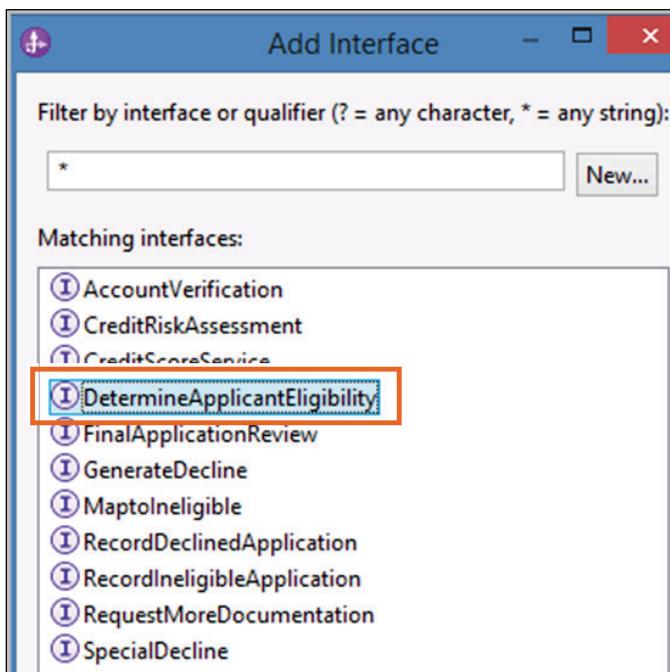
 1. Right-click RecordDeclinedApplication and click Wire to existing from the menu.
 2. The Java component is automatically wired to the **RecordDeclinedApplicationPartner** reference. To verify this connection, hover the mouse over the reference and the name that is displayed in a dialog box.



3. Save your changes.
6. Add a Java component that is named **DetermineApplicantEligibility** to the FoundationModule assembly diagram.

 1. In the assembly editor palette, expand **Components** and select **Java**.
 2. Click any blank space on the diagram to add the component.
 3. While the component is selected, switch to the **Description** tab in the **Properties** view.
 4. Change the **Name** to: **DetermineApplicantEligibility**
You will add the **DetermineApplicantEligibility** interface to the component.
 5. Select the **DetermineApplicantEligibility** component, and click the **Add Interface** icon in the action bar.

6. In the Add Interface dialog box, select the **DetermineApplicantEligibility** interface.



7. Click **OK**.
7. You will use the snippet code in `C:\labfiles\Support Files\Ex7\`
DetermineApplicantEligibilityImpl_InputCriterion.txt to generate the implementation for the `DetermineApplicantEligibility` component.

The snippet code returns values for each of the four test cases: IBM, ACME, TestCo, and AbcCo. For IBM, TestCo, and ACME, `eligibleApplication` is true by default. For AbcCo, `eligibleApplication` is false.

1. Right-click the `DetermineApplicantEligibility` component and click **Generate Implementation** from the menu.
2. In the Generate Implementation dialog box, select **com.education** and click **OK** to open `DetermineApplicantEligibilityImpl.java` in the Java editor.
3. Click the plus symbol to expand the `import` section and add the following two import statements:

```
import java.text.DateFormat;
import java.util.Date;
```

```

*FoundationModule - Assembly Diagram *DetermineApplicantEligibilityImpl.java X
package com.education;

import commonj.sdo.DataObject;
import com.ibm.websphere.sca.ServiceManager;
import java.text.DateFormat;
import java.util.Date;

```

4. In DetermineApplicantEligibilityImpl.java, browse to the public DataObject InputCriterion declaration at the end of the file.
 5. In Windows Explorer, browse to C:\labfiles\Support Files\Ex7\ and open DetermineApplicantEligibility_InputCriterion.txt in a text editor.
 6. Copy the text in DetermineApplicantEligibility_InputCriterion.txt and paste it over the green comments section (the text that begins with //) in the InputCriterion method.
- Important:** Remove `return null;` from the end of the InputCriterion method.
7. Alternatively, enter the following code for the InputCriterion method:

```

System.out.println( "[Java] Determine Applicant
Eligibility
begins" );

String date =
DateFormat.getDateInstance(DateFormat.MEDIUM).format( new Date() );


if
(credappin.getString( "companyName" ).equals( "ACME" ) )
{ credappin.setString( "applicationDate" , date );
credappin.setString( "companyName" , "ACME" );

credappin.setString( "customerCity" , "Berlin" );
credappin.setString( "customerCountry" , "Germany" );
credappin.setString( "contactFirstName" , "Torsten" );
credappin.setString( "contactLastName" , "Frings" );
credappin.setString( "contactPhoneNumber" , "905-555-7234" );
credappin.setInt( "requestAccountAmount" , 10000 );
credappin.setInt( "creditScore" , 0 );
credappin.setString( "productName" , "Labels" );
credappin.setInt( "pricingCode" , 23 );
credappin.setString( "comments" , "None" );
credappin.setString( "creditRating" , "C+" );
credappin.setString( "pricingScore" , "17" );

```

```
credappin.setBoolean("creditReportNeeded", true);
credappin.setString("creditRisk", "MED");
credappin.setBoolean("applicationDecision", true);
credappin.setBoolean("eligibleApplication", true);
credappin.setString("accountNumber", "ACM002");

} else if
(credappin.getString("companyName").equals("AbcCo")) {
credappin.setString("applicationDate", date);
credappin.setString("companyName", "AbcCo");
credappin.setString("customerCity", "Madrid");
credappin.setString("customerCountry", "Spain");
credappin.setString("contactFirstName", "Fernando");
credappin.setString("contactLastName", "Torres");
credappin.setString("contactPhoneNumber", "312-555-9725");

credappin.setInt("requestAccountAmount", 20000);
credappin.setInt("creditScore", 0);
credappin.setString("productName", "Tacks");
credappin.setInt("pricingCode", 51);
credappin.setString("comments", "Bad credit");
credappin.setString("creditRating", "F");
credappin.setString("pricingScore", "31");
credappin.setBoolean("creditReportNeeded", true);
credappin.setString("creditRisk", "HIGH");
credappin.setBoolean("applicationDecision",
false);
credappin.setBoolean("eligibleApplication",
false); credappin.setString("ineligibleReason",
"Bad credit");
credappin.setString("accountNumber", "ABC001");

} else if (credappin.getString("companyName").equals("IBM"))
{ credappin.setString("applicationDate", date);
credappin.setString("companyName", "IBM");
credappin.setString("customerCity", "Boston");
credappin.setString("customerCountry", "USA");
credappin.setString("contactFirstName", "Landon");
credappin.setString("contactLastName", "Donovan");
credappin.setString("contactPhoneNumber", "547-555-3172");

credappin.setInt("requestAccountAmount", 30000);
credappin.setInt("creditScore", 0);
credappin.setString("productName", "Pens");
credappin.setInt("pricingCode", 34);
credappin.setString("comments", "None");
```

```
        credappin.setString("creditRating", "A++");
        credappin.setString("pricingScore", "32");
        credappin.setBoolean("creditReportNeeded", true);
        credappin.setString("creditRisk", "LOW");
        credappin.setBoolean("applicationDecision", true);
        credappin.setBoolean("eligibleApplication", true);
        credappin.setString("accountNumber", "IBM007");

    } else if
        (credappin.getString("companyName").equals("TestCo")) {
        credappin.setString("applicationDate", date);
        credappin.setString("companyName", "TestCo");
        credappin.setString("customerCity", "Chicago");
        credappin.setString("customerCountry", "USA");
        credappin.setString("contactFirstName", "Jane");
        credappin.setString("contactLastName", "Doe");
        credappin.setString("contactPhoneNumber", "872-555-9915");
        credappin.setInt("requestAccountAmount", 50000);
        credappin.setInt("creditScore", 0);
        credappin.setString("productName", "Chairs");
        credappin.setInt("pricingCode", 57);
        credappin.setString("comments", "None");
        credappin.setString("creditRating", "C");
        credappin.setString("pricingScore", "11");
        credappin.setBoolean("creditReportNeeded", true);
        credappin.setString("creditRisk", "HIGH");
        credappin.setBoolean("applicationDecision", false);
        credappin.setBoolean("eligibleApplication", true);
        credappin.setString("accountNumber", "TEST001");
    }

    System.out.println("[Java] Determine Applicant
Eligibility
ends");

    return credappin;
```

Note: DetermineApplicantEligibilityImpl.java sets the CustomerApplication data based on the value in the companyName field. If the companyName is ACME, the output CustomerApplication data has the following values:

- applicationDate: date
- companyName: ACME
- customerCity: Berlin
- customerCountry: Germany
- contactFirstName: Torsten
- contactLastName: Frings
- contactPhoneNumber: 905-555-7234
- requestAccountAmount: 10000
- creditScore: 0
- productName: Labels
- pricingCode: 23
- comments: None
- creditRating: C+
- pricingScore: 17
- creditReportNeeded: true
- creditRisk: MED
- applicationDecision: true
- eligibleApplication: true
- accountNumber: ACM002

If the companyName is AbcCo, the output CustomerApplication data has the following values:

- applicationDate: date
- companyName: AbcCo
- customerCity: Madrid
- customerCountry: Spain
- contactFirstName: Fernando
- contactLastName: Torres
- contactPhoneNumber: 312-555-9725
- requestAccountAmount: 20000
- creditScore: 0
- productName: Tacks
- pricingCode: 51
- comments: Bad credit
- creditRating: F
- pricingScore: 31
- creditReportNeeded: true
- creditRisk: HIGH
- applicationDecision: false
- eligibleApplication: false (Note: the value is false for AbcCo.)
- ineligibleReason: Bad credit
- accountNumber: ABC001

If the `companyName` is `IBM`, the output `CustomerApplication` data has the following values:

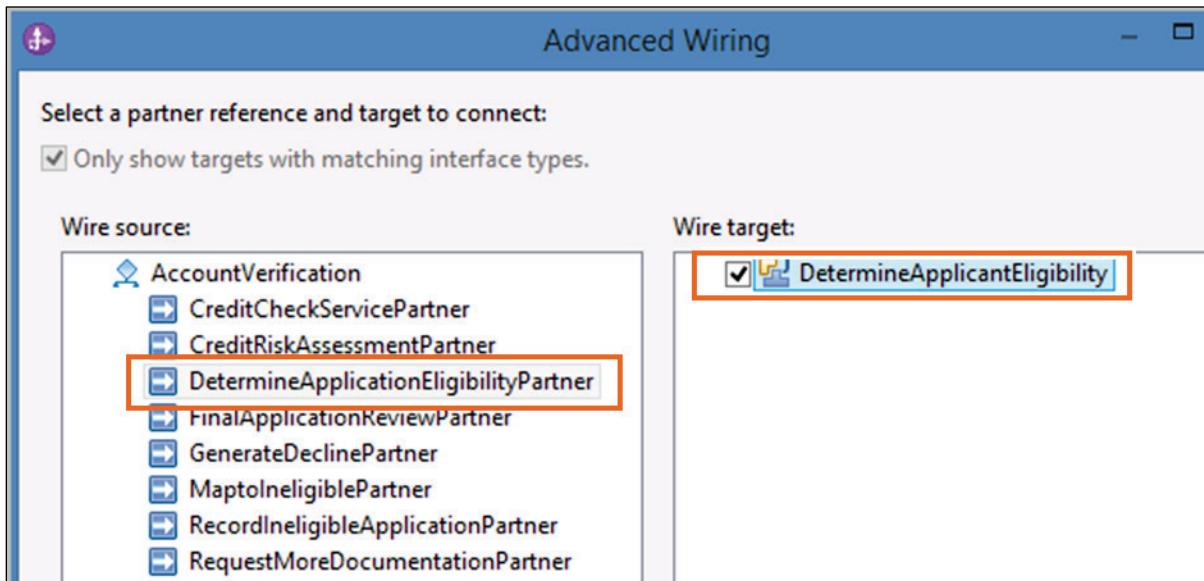
- `applicationDate`: date
- `companyName`: IBM
- `customerCity`: Boston
- `customerCountry`: USA
- `contactFirstName`: Landon
- `contactLastName`: Donovan
- `contactPhoneNumber`: 547-555-3172
- `requestAccountAmount`: 30000
- `creditScore`: 0
- `productName`: Pens
- `pricingCode`: 34
- `comments`: None
- `creditRating`: A++
- `pricingScore`: 32
- `creditReportNeeded`: true
- `creditRisk`: LOW
- `applicationDecision`: true
- `eligibleApplication`: true
- `accountNumber`: IBM007

If the companyName is TestCo, then the output CustomerApplication data has the following values:

- applicationDate: date
 - companyName: TestCo
 - customerCity: Chicago
 - customerCountry: USA
 - contactFirstName: Jane
 - contactLastName: Doe
 - contactPhoneNumber: 872-555-9915
 - requestAccountAmount: 50000
 - creditScore: 0
 - productName: Chairs
 - pricingCode: 57
 - comments: None
 - creditRating: C
 - pricingScore: 11
 - creditReportNeeded: true
 - creditRisk: HIGH
 - applicationDecision: false
 - eligibleApplication: true
 - accountNumber: TEST001
8. Press Ctrl+S to save your changes and Ctrl + W to close the Java editor.
 9. Close `DetermineApplicantEligibility_InputCriterion.txt` and close Windows Explorer.

8. Wire the DetermineApplicantEligibility Java component to DetermineApplicationEligibilityPartner of the AccountVerification process.

1. Right-click **AccountVerification** and select **Wire (Advanced)** from the menu.
2. In the “Wire source” window of the Advanced Wiring dialog box, select **DetermineApplicationEligibilityPartner**.
3. In the “Wire target” window, select the **DetermineApplicantEligibility** Java component.

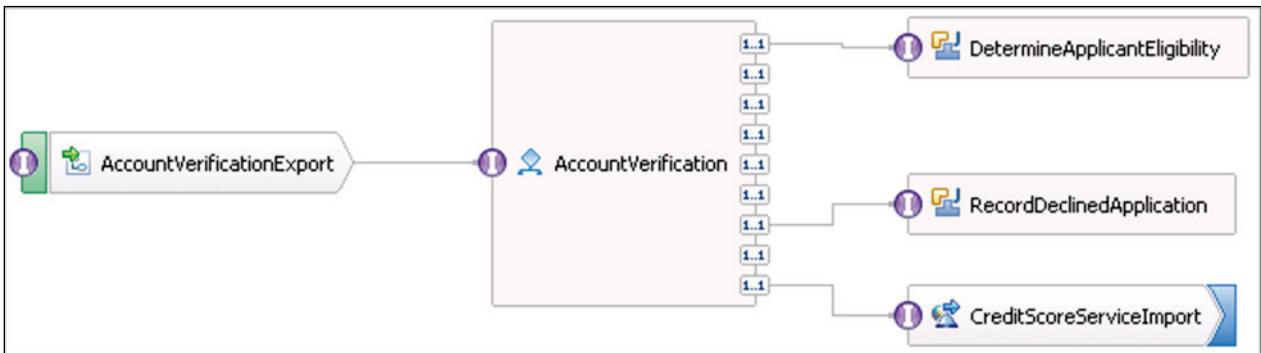


4. Click **OK**.
 5. Verify the wiring by hovering the mouse over the reference. The dialog box indicates that the component is wired to the **DetermineApplicationEligibilityPartner** reference.
Note: When you add the AccountVerification component to the FoundationModule assembly diagram, the order of the attached reference partners might differ from the diagram pictured. This difference does not indicate a problem. The order of the reference partners on the assembly diagram component does not reflect the order in which the process calls the reference partners.
 6. Save your changes.
9. Wire the CreditScoreServiceImport component to the CreditCheckServicePartner reference of the AccountVerification process.
1. Right-click CreditScoreServiceImport and click **Wire to Existing** from the menu.

- Verify the wiring by hovering the mouse over the reference. The dialog box indicates that the component is wired to the **CreditCheckServicePartner** reference.



- Save the assembly diagram. It resembles the following figure:



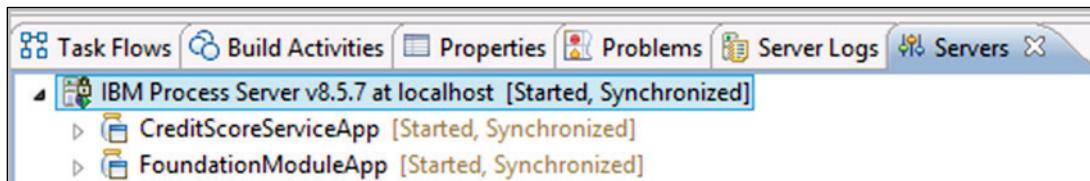
Task 4. Test a business process in the IBM Integration Designer test environment.

In this portion of the exercise, you test the FoundationModule (containing your business process) in the integrated test client. Services that are not yet implemented are emulated.

- Start the server (if it is not already running) and deploy FoundationModuleApp and CreditScoreServiceApp to the server.
 - Start the process server, if necessary.
 - Right-click **IBM Process Server v8.6 at localhost** and click **Add and Remove**.
 - Click **Add All** and click **Finish**.
 - Wait until the modules are published and started. Publishing is complete when no messages appear in the IBM Integration Designer status bar such as Publishing FoundationModuleApp.

The applications are started when you see the messages Application started: FoundationModuleApp and Application started: CreditScoreServiceApp in the **Server Logs** view.

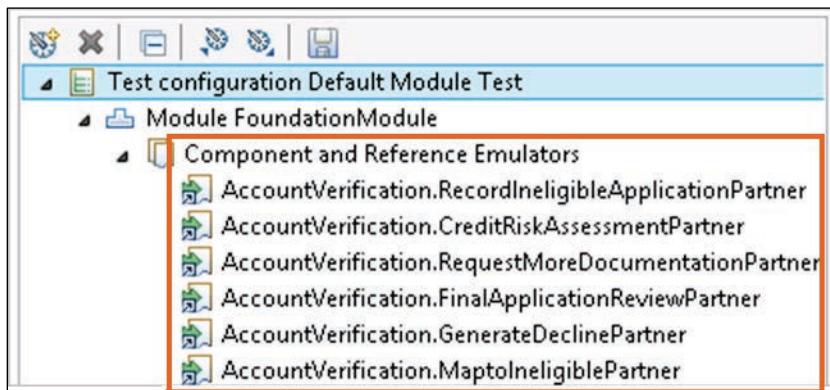
You can also verify the status of the modules by expanding the server.



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**. Continue to the next step when the status is changed to **Started**.

2. Use the Test Component option to test the AccountVerification process.

1. In the assembly diagram, right-click **AccountVerification**, and click **Test Component** from the menu.
2. When the integrated test client opens, switch to the **Configurations** tab. The unwired references of the business processes are emulated.

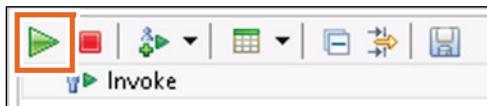


3. Switch to the **Events** tab.
4. In the **Initial request parameters** section, right-click **Input** and click **Import from File** from the menu.
5. In the “Import from File” dialog box, browse to C:\labfiles\Support Files\Ex7\.

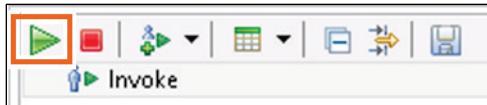
6. Select **EX7_Test_Data.xml** and click **Open** to populate the input parameters with the required test data.
7. Alternatively, you can manually enter the following values:
 - accountNumber: IBM007
 - applicationDate: 6/20/2014
 - applicationDecision: true
 - comments: None
 - companyName: IBM
 - contactFirstName: Landon
 - contactLastName: Donovan
 - contactPhoneNumber: 547-555-3172
 - creditRating: A++
 - creditReportNeeded: true
 - creditRisk: LOW
 - creditScore: 0
 - customerCity: Boston
 - customerCountry: USA
 - eligibleApplication: true
 - ineligibleReason: None
 - pricingCode: 34
 - pricingScore: 32
 - productName: Pens
 - requestAccountAmount: 30000

Name	Type	Value
Input	CustomerApplication	[ab]
accountNumber	string	[ab] IBM007
applicationDate	string	[ab] 11/22/2009
applicationDecision	boolean	[ab] true
comments	string	[ab] None
companyName	string	[ab] IBM
contactFirstName	string	[ab] Landon
contactLastName	string	[ab] Donovan
contactPhoneNumber	string	[ab] 547-555-3172
creditRating	string	[ab] A++
creditReportNeeded	boolean	[ab] true
creditRisk	string	[ab] LOW
creditScore	int	[ab] 0
customerCity	string	[ab] Boston
customerCountry	string	[ab] USA
eligibleApplication	boolean	[ab] true
ineligibleReason	string	[ab] None
pricingCode	string	[ab] 34
pricingScore	string	[ab] 32
productName	string	[ab] Pens
requestAccountAmount	int	[ab] 30000

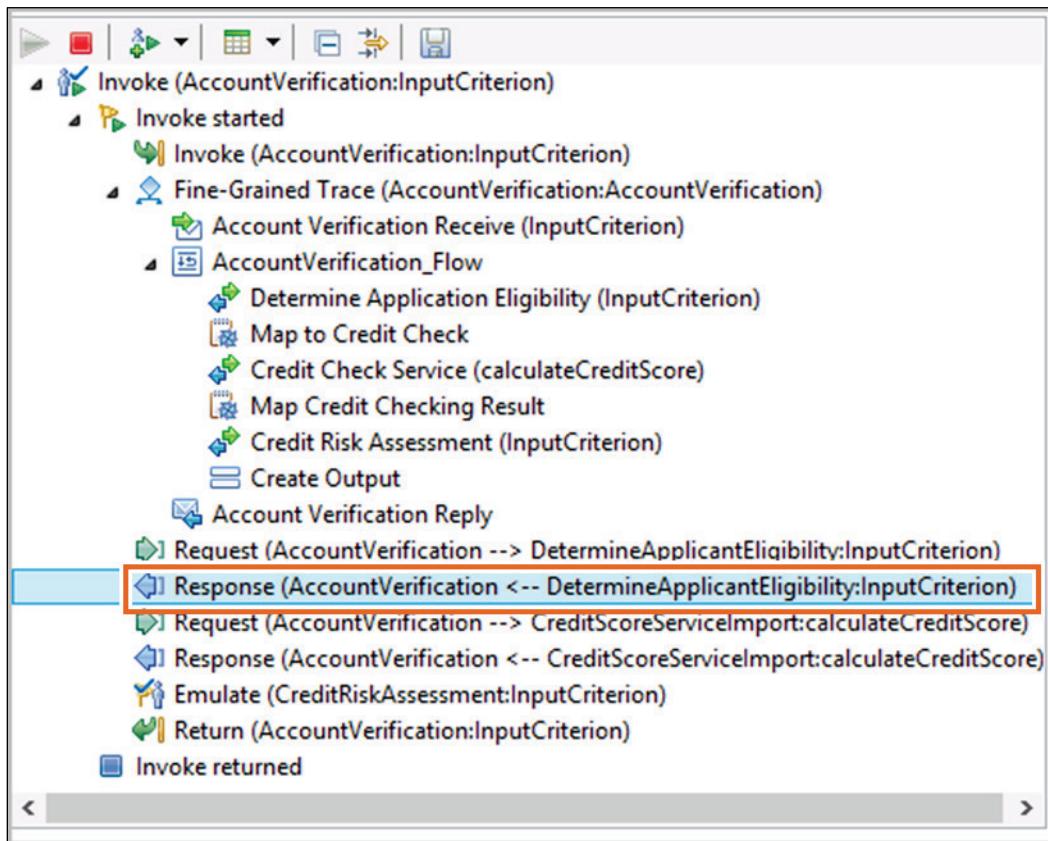
8. Click **Continue** on the Events toolbar to run the test.



9. For the deployment location, select **IBM Process Server v8.6 at localhost** and click **Finish**.
10. At the User Login dialog box, accept the default values for **User ID** and **Password** and click **OK**.
11. The test pauses at **Emulate (CreditRiskAssessment:InputCriterion)**; click **Continue** in the Events toolbar to resume the test.

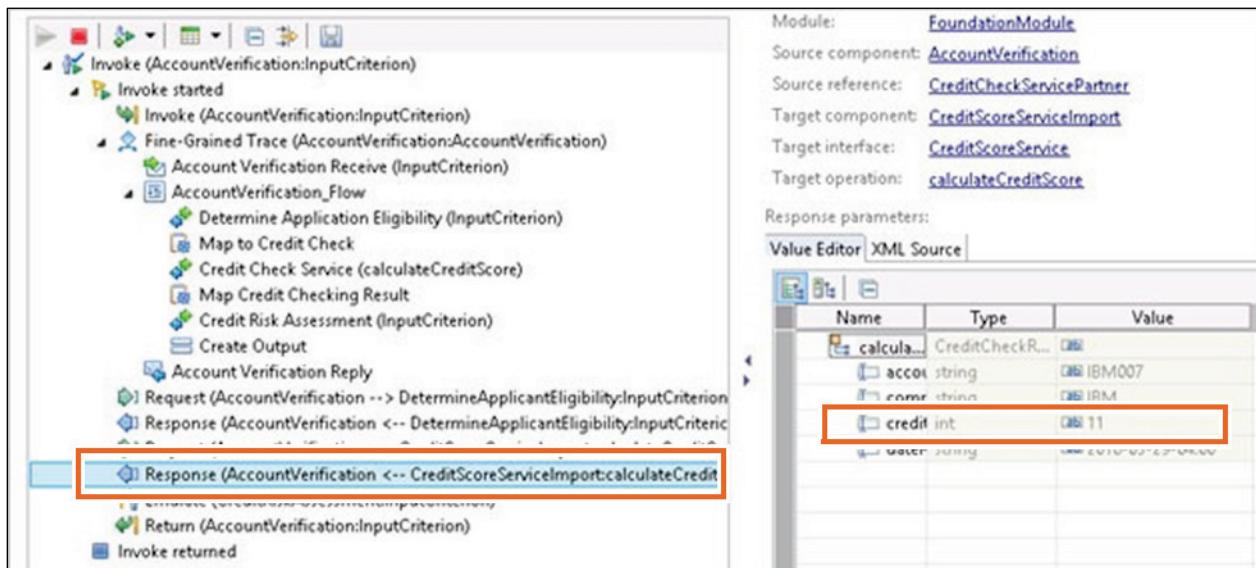


12. Notice the first response that is returned from the **DetermineApplicantEligibility** service.

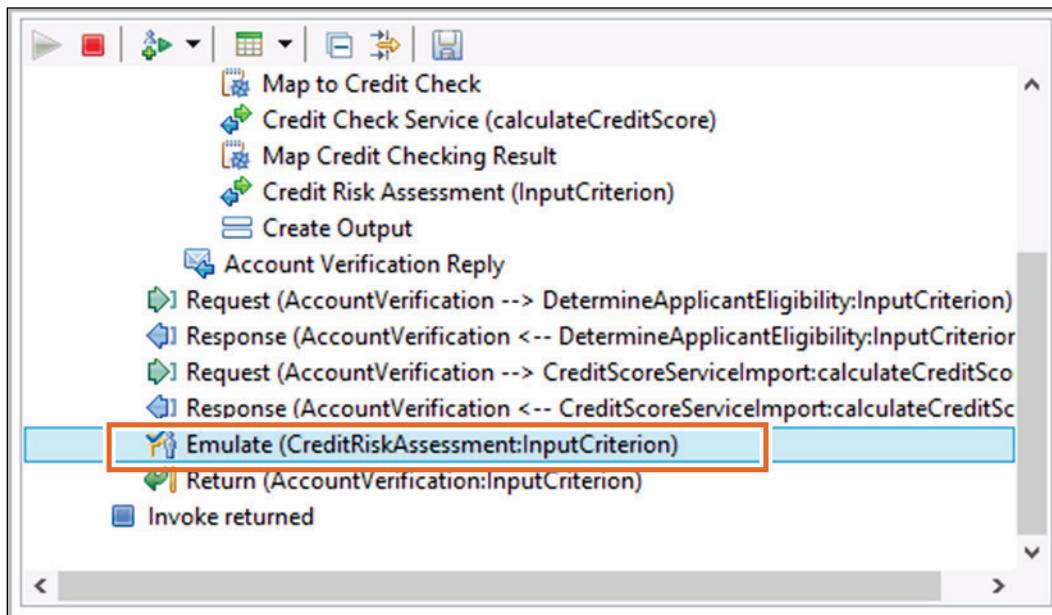


Based on the response from the **DetermineApplicantEligibility** service, the process takes either the eligible application path or the ineligible application path. In three of the four test cases (IBM, TestCo, and ACME), **eligibleApplication** is set to true. If the **companyName** is set to **AbcCo**, the **DetermineApplicantEligibility** service returns an **eligibleApplication** value of false. Because your test data uses **IBM** for the **companyName**, the eligible application path is followed.

13. Notice the second response that is from the CreditScoreService. If the application is eligible, the data is transformed and sent to the CreditScoreService. Because your test data was for IBM, the web service returns a credit score of 11. This score is a high credit score and represents a low credit risk. Select the **Response (AccountVerification <- CreditScoreServiceImport:calculateCreditScore)** event and examine the data in the **Response parameters** section. A creditScore of 11 is returned.



14. The test stops at the first service emulation. Because the CreditRiskAssessment service is not implemented, the service is emulated and requires you to manually send the response data back to the business process. Select **Emulate (CreditRiskAssessment:InputCriterion)**.

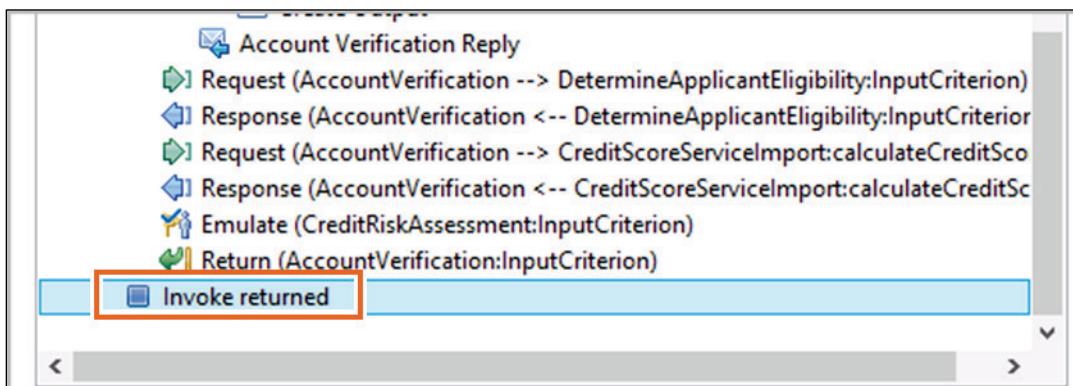


15. In the **Output parameters** section, scroll to the **creditRisk** attribute.

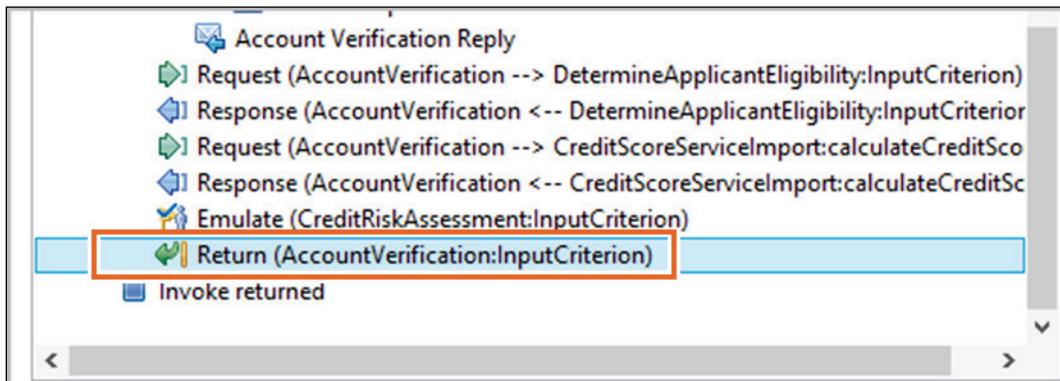
16. Verify that **LOW** is displayed in the **Value** column.

Name	Type	Value
contactLastName	string	[ab] Donovan
contactPhoneNum	string	[ab] 547-555-3172
creditRating	string	[ab] A++
creditReportNeeded	boolean	[ab] true
creditRisk	string	[ab] LOW
creditScore	int	[ab] 11
customerCity	string	[ab] Boston
customerCountry	string	[ab] USA

17. The blue, square **Invoke returned** stop node indicates completion of the test trace.



18. Because the **creditRisk** was **LOW**, the application was automatically approved. Select the **Return (AccountVerification:InputCriterion)** event.



19. Examine the output in the **Return parameters** section.

Name	Type	Value
Output	Message	Risk was LOW. Application automatically approved.
message	string	Risk was LOW. Application automatically approved.

20. If you are prompted to save the test trace, close the **FoundationModule_Test** tab and click **No**.

Note: In future exercises, as you implement the remaining services that your application uses, you use the prebuilt test cases to test other paths through the business process.

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. Close IBM Integration Designer.

Results:

In this exercise, you completed a complex business process and tested it in the IBM Integration Designer test environment.

References

- OASIS WS-BPEL Technical Committee
 - http://www.oasis-open.org/committees/tc_home.php?wg_abbrev=wsbpel
- BPEL4WS and draft WS-BPEL specifications from IBM developerWorks
 - <http://bpel.xml.org/specifications>
- Business Process Choreographer samples
 - <http://publib.boulder.ibm.com/bpcsamp/v7r5/index.html>
- Versioning business processes and human tasks
 - http://www.ibm.com/developerworks/websphere/library/techarticles/0808_smolny/0808_smolny.html

Unit 11 Business rules

The slide features a blue header bar with 'IBM Training' on the left and the IBM logo on the right. The main content area has a light gray diagonal striped background. In the center, the title 'Business rules' is displayed in bold blue text. Below it, the text 'IBM Business Process Manager V8.6' is also in blue. At the bottom of the slide, there is a copyright notice: '© Copyright IBM Corporation 2018' and 'Course materials may not be reproduced in whole or in part without the written permission of IBM.'

IBM Training

Business rules

IBM Business Process Manager V8.6

© Copyright IBM Corporation 2018
Course materials may not be reproduced in whole or in part without the written permission of IBM.

Unit objectives

- Define the purpose and business value of using business rules
- Describe the function of a rule group and list the rule group components
- Define the concepts of rule sets and decision tables
- Describe the runtime behavior of a rule group component
- Identify the IBM Process Server administrative capabilities for importing, exporting, and auditing business rule changes in the runtime environment

Topics

- Overview of business rules
- Runtime aspects of rule groups

Business rules

© Copyright IBM Corporation 2018

Topics

Overview of business rules

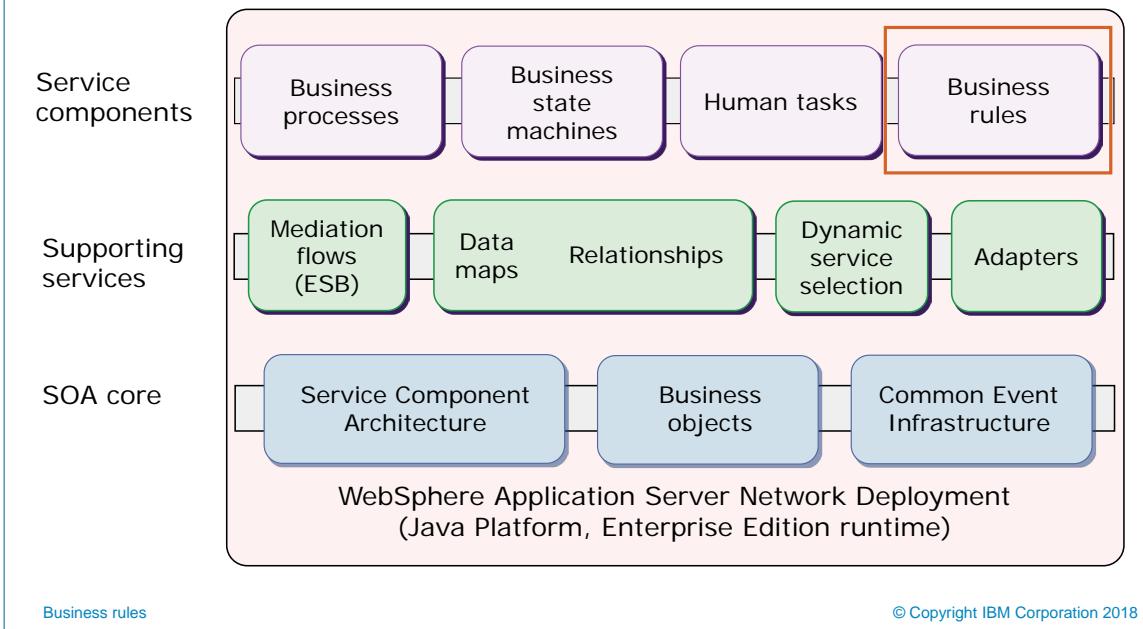
Business rules

© Copyright IBM Corporation 2018

Overview of business rules

Business rules are a service component

- Business rules are part of the service components layer

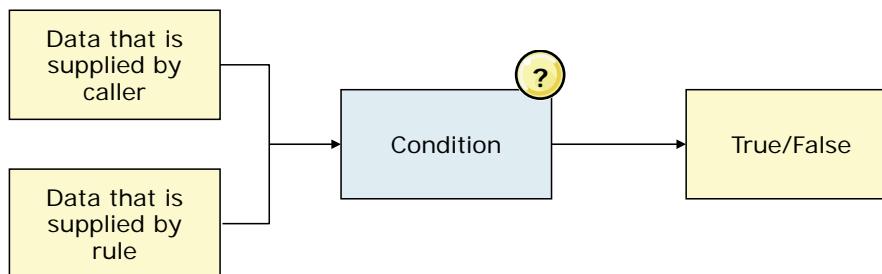


Business rules are a service component

Business rules externalize frequently altered business policies. Business rules are an implementation type that is exposed as an SCA component. As an SCA component, business rules are exposed as services and can be called like any other SCA component.

What is a business rule?

- A business rule captures and implements business policies and practices by using one or more if-then statements
 - For example: If $orderTotal >= 1000$, then the discount = 0.10
- A business rule consists of a condition (an expression that uses data that the caller and the rule supply) and one or more actions
 - The condition is the “if” portion of the statement
 - Evaluation of the condition is either true or false
 - The action is the “then” portion of the statement



Business rules

© Copyright IBM Corporation 2018

What is a business rule?

When a business rule runs, a series of conditions are evaluated and a series of actions done. The outcome of the evaluation of the condition is a true or false value. An action is a simple piece of logic that can invoke a service, update a local variable, or as is most commonly the case, modify data that is returned to the caller. The action has access to both data supplied by the caller and data that is supplied by the rule.

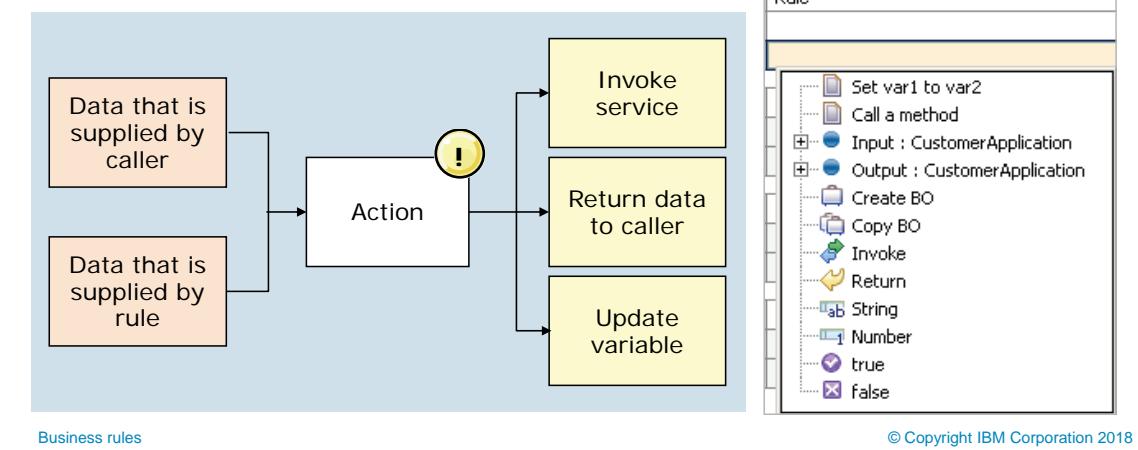
A business rule is used to help abstract the client from implementation of the business logic. A calling client does not care how a business rule carries out its duties; it sends in the inputs and expects a result. Organization of rules is covered next, but to set the stage, business rules are assembled into business rule groups and can be implemented in one of two ways, rule sets or decision tables.

Examples of business rules include:

- A rental car company is able to change corporate discount amounts during peak dates and times: summer and holidays.
- An insurance agency can invoke a process for an insurance claim that happened in the past. It can happen because of a company merger or through a change in company policy.
- Shipping rates might change based on destination, package weight and size, or delivery priorities (overnight or next day).

Business rule actions

- Possible actions that result from the evaluation of the condition:
 - Invoking a service
 - Updating a local variable
 - Modifying data that is going to be returned to the caller
 - Creating or copying a business object
 - Returning (stopping execution of the rules early)

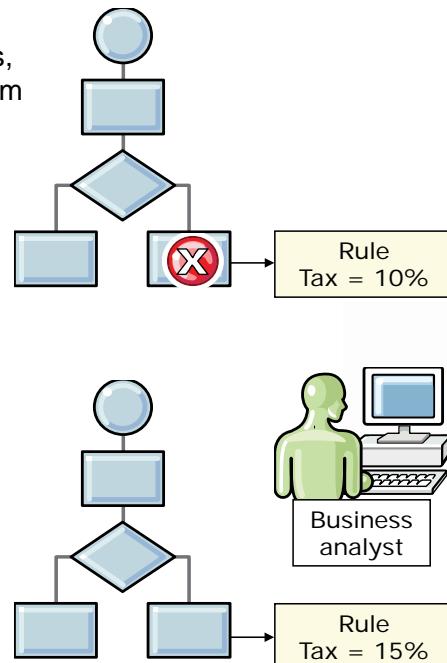


Business rule actions

When the return statement runs, evaluation of rules stops. Output variable values at the return point become the output of the rules.

Business value of rules

- By using rule groups to expose rules as services, rules are separated from processes that use them
 - Multiple business processes can use the same sets of business rules
 - Rules are no longer in application code
- The business analyst can quickly change exposed rules at run time
 - If rules are not in application code, you are no longer bound to IT development cycles
 - The developer is needed only for more complex changes
- Rule groups are SCA components
 - As an SCA implementation type, it abstracts and decouples the rule implementation
- Two ways to implement business rules
 - Rule sets
 - Decision tables



Business rules

© Copyright IBM Corporation 2018

Business value of rules

Hardcoding values and decision points into a business process is undesirable. Business values that are subject to change would result in manual recoding of the process and redeployment of the solution. These manual activities are error prone and take lot of time. IBM Process Server provides a much more elegant solution through business rules.

Decisions or values that affect the operation of a process can be externalized outside of the process. When these values are needed, the process asks for these values in a dynamic fashion by invoking the business rule service. IBM Process Server maintains the business rule values in a secure data store. They can be designed, configured, or modified at a high level through a specialized administrative web client. In principle, suitable staff members can modify these values without having to know anything about the underlying technical characteristics of the solution.

Why do you need a business rule service component when BPEL has a choice activity? In BPEL, you can use activities such as choice or receive choice; however, you are hardcoding the decision-making rules in the application. In a hardcoded logic, changes cannot be made easily or quickly.

By having a business rules component, the decision-making logic can be separated from the business process. Multiple services can invoke the rules, and business rules become reusable.

Changes in business rules (for example, the discount rate changes from 10% to 12%) do cause changes in business processes.

Business rules help abstract the implementation (the actual work done) from the client (the object that calls the work). Abstraction allows a company to change, update, and reuse code easily. This model of abstraction fits naturally within loosely coupled SOA processes. When they are decoupled, it is easy for other SCA services to access business rules. For example:

- An analyst can change loan approval criteria without redeploying the application.
- Flexibility allows for quick responses to changing business conditions and customer demand.
- Rules are reusable: multiple business processes can use the same set of business rules.

Rule sets defined

- Two types of rules can be used in a rule set: if-then and action
 - An if-then rule evaluates a condition, and then performs one or more actions
 - An action rule always performs actions, regardless of the input, without using a condition
- All rules in a rule set are evaluated sequentially, first to last
- Each **condition** that evaluates to true causes an **action** to be performed
 - More than one rule can be fired
- If-then and action rules can be exposed as template rules
 - Templates are exposed at run time in a natural language format
- The Business Rules Manager web client or Business Space client can be used to edit template rule parameters at run time
 - The developer can constrain parameter values

Rule sets defined

Business rules can be implemented in two ways: rule sets and decision tables. A rule set is a set of one or more if-then condition or action statements that are evaluated sequentially. Therefore, the first business rule that is listed is going to be the first one evaluated when called at run time. The second rule is evaluated next, continuing until the last one is evaluated. Rule sets hold any number of action or if-then conditions. A rule set can evaluate multiple conditions and can process multiple rules.

Rule sets in IBM Integration Designer (1 of 2)

- Rule sets in IBM Integration Designer contain:
 - Name (and Display Name)**
 - Interface**: Provides the operation, inputs, and outputs
 - Variables**: Rule actions can update local variables that you define

The screenshot shows the 'Rule Set' editor interface. It is divided into three main sections:

- 1 Rule Set**: Displays the rule set name 'checkCustomerTypeRS' and its display name.
- 2 Interface**: Shows the interface details, including the interface name 'CheckCustomerType', operation 'checkCustomerType', input 'inputCustomerType' (string type), and output 'outputValidType' (boolean type).
- 3 Variables**: A table for defining local variables, currently empty.

(Rule set contents continue on the following page)

Business rules © Copyright IBM Corporation 2018

Rule sets in IBM Integration Designer

The areas of the rule set editor in the figure are as follows:

1. The rule set properties: Use this icon to see detailed properties for your rule set.
2. The interface area: This area displays the interface that the rule set currently references.
3. The variables area: This area displays the variables that the rule set uses to store its data. To create a variable, click the icon; or to remove one, highlight it and click the icon.

Rule sets in IBM Integration Designer (2 of 2)

- Rule sets in IBM Integration Designer also contain:
 4. **Rules:** If-then rules and action rules
 5. **Templates:** Rules with editable parameters that are exposed in the runtime environment

The screenshot shows the IBM Integration Designer interface with two main panes: 'Rules' and 'Templates'.

Rules Pane (4):

- Toolbar icons:** Create If-Then Rule, Create Action Rule, Create Template Rule, Convert Rule to a Template.
- Table:**

Name	Rule1
Template	Template_Rule1
Presentation	If inputCustomerType.equalsIgnoreCase(Competitor) == [false] then outputValidType = [true]
- Info Box:** The **Rules** icon palette includes:
▪ Add If-Then Rule
▪ Add Action Rule
▪ Add Template Rule
▪ Convert Rule to a Template

Templates Pane (5):

- Toolbar icons:** Create If-Then Template, Create Action Template.
- Table:**

Name	Template_Rule1												
Presentation	If inputCustomerType.equalsIgnoreCase(stringParam0) == booleanParam1 then outputValidType = booleanParam2												
Description													
Parameters	<table border="1"> <tr> <th>Name</th> <th>Type</th> <th>Constraint</th> </tr> <tr> <td>stringParam0</td> <td>string</td> <td>None</td> </tr> <tr> <td>booleanParam1</td> <td>boolean</td> <td>None</td> </tr> <tr> <td>booleanParam2</td> <td>boolean</td> <td>None</td> </tr> </table>	Name	Type	Constraint	stringParam0	string	None	booleanParam1	boolean	None	booleanParam2	boolean	None
Name	Type	Constraint											
stringParam0	string	None											
booleanParam1	boolean	None											
booleanParam2	boolean	None											
If	inputCustomerType.equalsIgnoreCase(stringParam0) == booleanParam1												
Then	outputValidType = booleanParam2												
- Info Box:** The **Templates** icon palette includes:
▪ Add If-Then Template
▪ Add Action Template
- Info Box:** Exposed parameters can be constrained to a predefined set of values

Business rules

© Copyright IBM Corporation 2018

The remaining portions of the rule set editor are:

4. The icons in the rules area toolbar, which have the following functions:
 - Create an if-then rule
 - Create an action rule
 - Create a template rule
 - Create a template that is based on the current rule
5. The templates area: Use this area to graphically compose templates for a rule set. The icons in the templates area toolbar have the following functions:
 - Create a template for an if-then rule
 - Create a template for an action rule

A constraint is a restriction that is placed in a template by the person who creates it. This constraint limits how much a specified parameter can be modified. The two main types of constraints are:

- Range constraints: Range constraints apply to numeric types that are used within rules. For example, an authorized business user is allowed to adjust the discount on some merchandise, but these changes must be within 5% – 30%.
- Enumeration constraints: Enumeration constraints take the form of a list of either a numeric or a string type. The authorized business user must choose from one of the options available in the list, for example, upgrading a customer's credit rating from "silver" to "gold."

Decision table

- Decision tables represent multiple rule conditions in a table
 - Each axis of the table is an “if” condition
 - A single “then” action is defined at the intersection of the conditions
 - Only one rule is fired that meets all the conditions
 - A special “otherwise” condition can be used for the case when none of the conditions are met
- An example table is shown:
 - Axes represent package weight and volume conditions
 - The action that is taken is the intersection of the weight and volume criteria
- Decision table rules can also be exposed as templates

The diagram illustrates a decision table structure. At the top, a box labeled "Weight condition" spans two columns of the table. Below the table, a box labeled "Volume condition" has an arrow pointing upwards towards the first row of the table. The table itself has a header row labeled "Conditions". The first column is labeled "package.weight" and contains rows for "< 1", "> 1 and < 5", and "> 5". The second column is labeled "package.volume" and contains rows for "< 9" and "> 9". The third column is labeled "Actions" and contains values "shippingAndHandlingCharge", "5", "7", and "7". The fourth column is labeled "Actions" and contains values "shippingAndHandlingCharge", "5", "7", and "7". The bottom right corner of the table area contains the text "© Copyright IBM Corporation 2018".

Conditions			
package.weight	< 1	> 1 and < 5	> 5
package.volume	shippingAndHandlingCharge	shippingAndHandlingCharge	shippingAndHandlingCharge
< 9	2	5	7
> 9	7	7	7
Actions			

Business rules

© Copyright IBM Corporation 2018

Decision table

The nested tree structure of a decision table evaluates multiple rules efficiently. The order of condition evaluation is not specified. If more than one alternative is possible, the choice is non-deterministic.

A decision table represents a multi-dimensional nested if-then structure. It is a rule set that can handle more complex decisions than a simple if-then decision. The decision table is a set of “if” conditions with “then” actions that are defined at the intersection points of the table. Conditions are evaluated in a nested order (navigation tree). A decision table evaluates one or more conditions, but processes only one rule.

Decision tables in IBM Integration Designer (1 of 2)

- Decision tables in IBM Integration Designer contain:
 - Name (and Display Name)**
 - Interface:** Provides operations, inputs, and outputs for the decision table

Decision Table	
Name	calcShipping
Display Name	
Interface	
Interface	CalculateShipping
Operation	calcShipping
Input	packageType
Output	shippingCost

(Decision table contents continue on the following page)

Decision tables in IBM Integration Designer

The areas of the decision table editor are as follows:

- The decision table properties: Use this icon to see detailed properties for your decision table.
- The interface area: This area displays the interface that is being referenced, and the inputs and outputs that you can use in the decision table.

Decision tables in IBM Integration Designer (2 of 2)

- Decision tables also contain:
 - Initialize:** An action rule that automatically works on an operation when data is first passed to the decision table, such as copying a business object or initializing a variable; only one initialize rule is allowed
 - Table:** Contains the multi-dimensional if-then rules

The screenshot shows the IBM Integration Designer interface for creating decision tables. It features two main sections: 'Initialize' (step 3) and 'Table' (step 4). The 'Initialize' section includes a toolbar with icons for creating initialization rules and converting them to templates. The 'Table' section includes a toolbar with icons for adding conditions, actions, and changing orientation, along with a 'Convert rule or cell to template' option. Below these sections is a decision table grid with four rows and four columns. The first row has conditions 'package.weight < 1', 'package.weight > 1 and < 5', and 'package.weight > 5'. The second row has conditions 'package.volume < 9' and 'package.volume > 9'. The third row has actions 'shippingAndHandlingCharge 2', 'shippingAndHandlingCharge 5', and 'shippingAndHandlingCharge 7'. The fourth row has actions 'shippingAndHandlingCharge 7', 'shippingAndHandlingCharge 7', and 'shippingAndHandlingCharge 7'. A legend at the bottom left identifies the icons: a blue plus sign for 'Add a condition', a green plus sign for 'Add a condition value', a grey plus sign for 'Add an action', and a blue square for 'Change orientation'.

Conditions	< 1	> 1 and < 5	> 5
package.weight			
package.volume	shippingAndHandlingCharge	shippingAndHandlingCharge	shippingAndHandlingCharge
< 9	2	5	7
> 9	7	7	7
Actions			

Business rules © Copyright IBM Corporation 2018

The areas of the decision table editor are continued:

- The initialize area: Use the icons in the initialize toolbar to do the following functions:

- Add an initialization action rule (an operation that takes place when data first enters a decision table)
- Create a template from this action rule

An initialization action rule automatically does an operation of some kind when data is first being passed to a decision table. Typically, an initialization action rule is used for the following reasons:

- To set an initial value, for example, to copy input business objects to output business objects.
- To initialize an output business object, for instance, to set the initial value of the business object attributes. More importantly, if the business object has attributes that are also business objects, then you can create those business objects (and this process can be recursive if they in turn contain even more business objects).

Initialization action rules can also be made into templates so that they can be modified in the runtime environment.

4. The decision table area: Use the icons in the decision table toolbar to have the following functions:

- Add a condition
- Add a condition value
- Add an action
- Change the orientation of the condition
- Create a template for one of the expressions in this table

The **conditions area** is displayed in light blue. Use this area of the decision table to define the multiple conditions that evaluate the incoming inputs to fire a corresponding action.

The **actions area** is displayed in light gray. Use this area of the decision table to define the actions that fire when the conditions intersect.

Rule sets versus decision tables

- Use rule sets when:
 - You want to fire multiple rules
 - You have several simple rules, each with one condition
- Use decision tables when:
 - You want to fire only one rule
 - You have complex rules with multiple conditions
- Rule sets and decision tables can be used together
 - You can have several rule sets
 - You can have several decision tables
 - You can have combinations of rule sets and decision tables

Rule sets versus decision tables

Use decision tables when you have rules with multiple clauses or variables in the condition statements. More importantly, use a decision table when you want to process only one rule.

Use rule sets when you have rules with a few clauses or variables in the condition statements, or you must process multiple rules. (Evaluating sequentially can be inefficient when many rules exist.)

A developer typically does not have the authority to make business decisions that involve rules (such as how much of a discount to give to certain types of customers). When you develop business rules, you must typically involve a business analyst.

Rule groups

- A rule group is an SCA component that is used to dynamically invoke rule sets and decision tables, which are based on set criteria
 - The criterion is a date and time range
 - Only one target runs based on a date selection criterion
 - Start and end date/time criteria, and target rule sets or decision tables, can be modified at run time
 - Criteria can also be based on the content of the business object input, but this type of choice must be created programmatically
 - New rule sets and decision tables cannot be created at run time, but new rules can be created from exposed templates
- Groups organize rules that share a common business purpose
 - Groups are searchable at run time
 - A rule group might contain any number of rule sets or decision tables
- Rule groups are presented like any other SCA component

Rule groups

A business rule group is the highest level implementation component for a business rule. The main idea behind a rule group is to gather rule sets or decision tables that share a common business focus. It also serves as the gateway to the business rules because it is exposed in the runtime environment as an SCA service. All other services invoke a business rule on a request that a client (BPEL, selector) sends through the rule group. Therefore, a rule group must be established before a rule set or decision table can run.

One of the most important functions of the rule group is to define a date and time range for business rule execution. For example, during the time between 1/1/05 12:00 AM and 1/1/06 12:00 AM, use decision table one. The date and time that are provided during invocation determine which business rule to use.

Rule implementations can be “scheduled” for a point in the past, present, or future. This scheduling gives a company the ability to invoke a process as it was in the past, or to switch over to the next version of a business rule on a future date.

Business rule groups can be searched. In each search data field, you can select one of four query operators: **is equal to**, **is like**, **is not equal to**, **is not like**. The **is like** and **is not like** operators can contain wildcard characters, such as the percent sign, to specify a wildcard for any number of characters and an underscore to specify a single wildcard character. The wildcards must follow SQL syntax.

You can also select from the user-defined properties to add to the search context, and you can combine these properties by logical operators “and,” “or” and “not.” When you add, delete or modify properties on the search page, it applies only to the search context. It does not affect the properties of any rule object inside the business rules manager.

Rule group components

- Rule groups are composed of:
 1. An interface
 2. A default destination (ensures a target)
 3. One or more target destinations corresponding to the start and end date/time ranges (dates cannot overlap)
 4. A date selection criterion (the value is compared to the target start-end dates)
 5. A set of available destinations (rule sets and decision tables)

The screenshot shows the IBM Business Rules interface with the following components highlighted:

- 1** CreditRiskAssessment interface, under Interfaces.
- 2** Default Rule Logic: CreditRiskAssessmentRS.
- 3** Scheduled Rule Logic table:

Start Date	End Date	Rule Logic
Jan 13, 2008 12:00 AM	Mar 8, 2008 12:00 AM	CreditRiskAssessmentRS
Mar 9, 2008 12:00 AM	Jul 1, 2008 12:00 AM	CreditRiskAssessmentDT
- 4** Selection Criteria: Current date.
- 5** Available Rule Logic table:

CreditRiskAssessmentRS	CreditRiskAssessmentDT
------------------------	------------------------

Business rules © Copyright IBM Corporation 2018

Rule group components

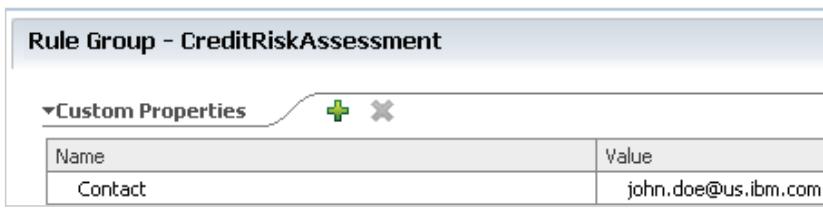
A rule group has five basic components:

1. An interface with an operation and operation parameters (for example, input1 and output1).
2. A default destination (a rule set or decision table to use when none of the active destinations apply).
3. A set of active destinations that are composed of rule sets and decision tables with a set of start and end date ranges for each of the active destinations.
4. A selection criterion that is used to evaluate which destination is used. The three criteria options are:
 - To use the current date and time (that is, the current date and time on the server that runs the application)
 - To “mine” for a date and time by using XPath or a Java snippet
 - To use custom Java code to return a date and time
5. A set of available destinations that can be added to the list of active destinations.

The example shows only one rule set and one decision table, but you can have multiples of only rule sets, or decision tables, or mixtures of both.

Business rule group properties

- Two types of rule group properties: system and user-defined
- System properties are read-only, and IBM Integration Designer and IBM Process Server use them internally
- User-defined (custom) properties are read and write
- Custom properties are used for the management of rule groups
 - Number of user properties that you can define is unlimited
 - Add and delete in IBM Integration Designer, and modify in Business Rules Manager
 - Can be queried through Business Rules Manager and the public business rules management API
 - Can hold customer-specific information



The screenshot shows a table titled "Rule Group - CreditRiskAssessment" under the "Custom Properties" section. The table has two columns: "Name" and "Value". A single row is present with the name "Contact" and the value "john.doe@us.ibm.com". There are "+" and "-" buttons at the top of the table for adding or removing rows. The bottom left corner of the screenshot area says "Business rules" and the bottom right corner says "© Copyright IBM Corporation 2018".

Business rule group properties

Two types of business rule group properties exist: system properties, which are IBM-use only and read-only; and user-defined or “custom” properties, which are read/write. Custom property names and values are case-sensitive.

Business rule group properties are defined in IBM Integration Designer but are modifiable in the business rules manager web client.

You can use custom properties on business rule groups for searches to retrieve subsets of business rule groups that you want to view and modify. You add new custom properties, delete existing properties, or modify existing properties through the editing pages of business rule groups. The number of custom properties on a business rule group is unlimited.

Display names and description fields for rule groups

- The **Description** field is for rules, rule sets, decision tables, and rule groups
- The **Display Name** field is for rule sets, decision tables, and rules that are used to support documentation
- Display names:
 - Can be any string value with special characters
 - Do not have to be unique

General Information

Display Name	CreditRiskAssessmentRS	<input checked="" type="checkbox"/> Synchronize with the name
Last Published	Jan 24, 2008 13:00 (Local Time)	Status Original
Description	<input type="text"/> <div style="border: 1px solid black; padding: 5px; width: 100%;">You can add a description for the rule set and for individual rules</div>	

Rules

Name	Display Name	Rule	Description	Action
RiskHIGH	RiskHIGH	If the customer credit score is greater than <input type="text" value="0"/> and less than <input type="text" value="4"/> then the credit risk is <input type="text" value="HIGH"/>	<input type="text"/>	<input type="button" value="Delete"/> <input checked="" type="checkbox"/> Synchronize Name

Business rules

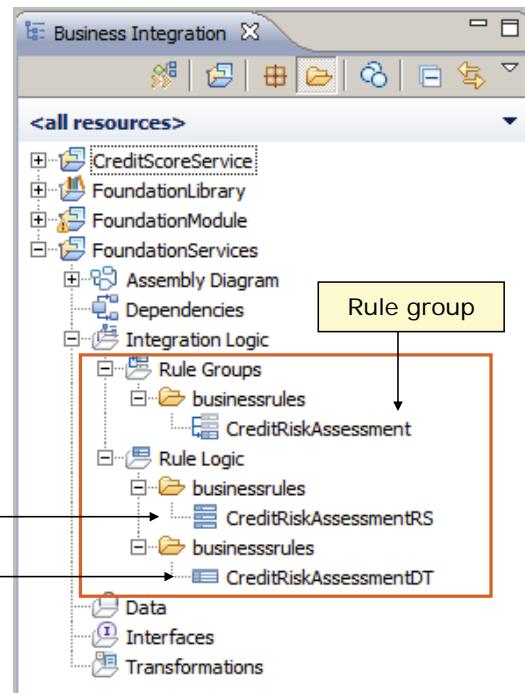
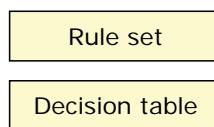
© Copyright IBM Corporation 2018

Display names and description fields for rule groups

The display names replace the current name values of rule groups, and they are used everywhere that the name values are currently shown, including the left navigator and when the artifacts are displayed in detail. The display names for rule groups are not required to be unique inside the business rule artifacts of the same type (business rule groups). Names of the business rule artifacts are still required to be unique in any use cases.

Business rules: Tools (1 of 2)

- IBM Integration Designer tools:
 - Easy to use tools for defining, executing, and managing business rules
 - Eclipse based tools for Business Rule development
 - Focused on the more technical developer role
 - Business Integration view is used mostly for Business Rule interaction



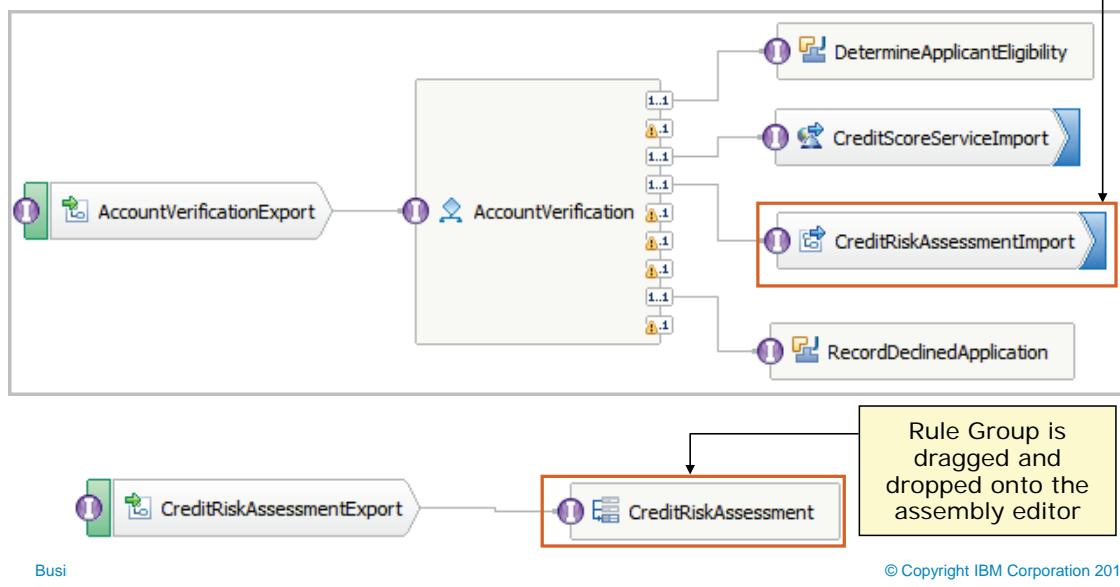
© Copyright IBM Corporation 2018

Business rules: Tools

Business rules: Tools (2 of 2)

- Assembly editor is used to connect Business rules to calling SCA Components

Import used to connect Business rule to another module



Business rules in IBM Process Designer

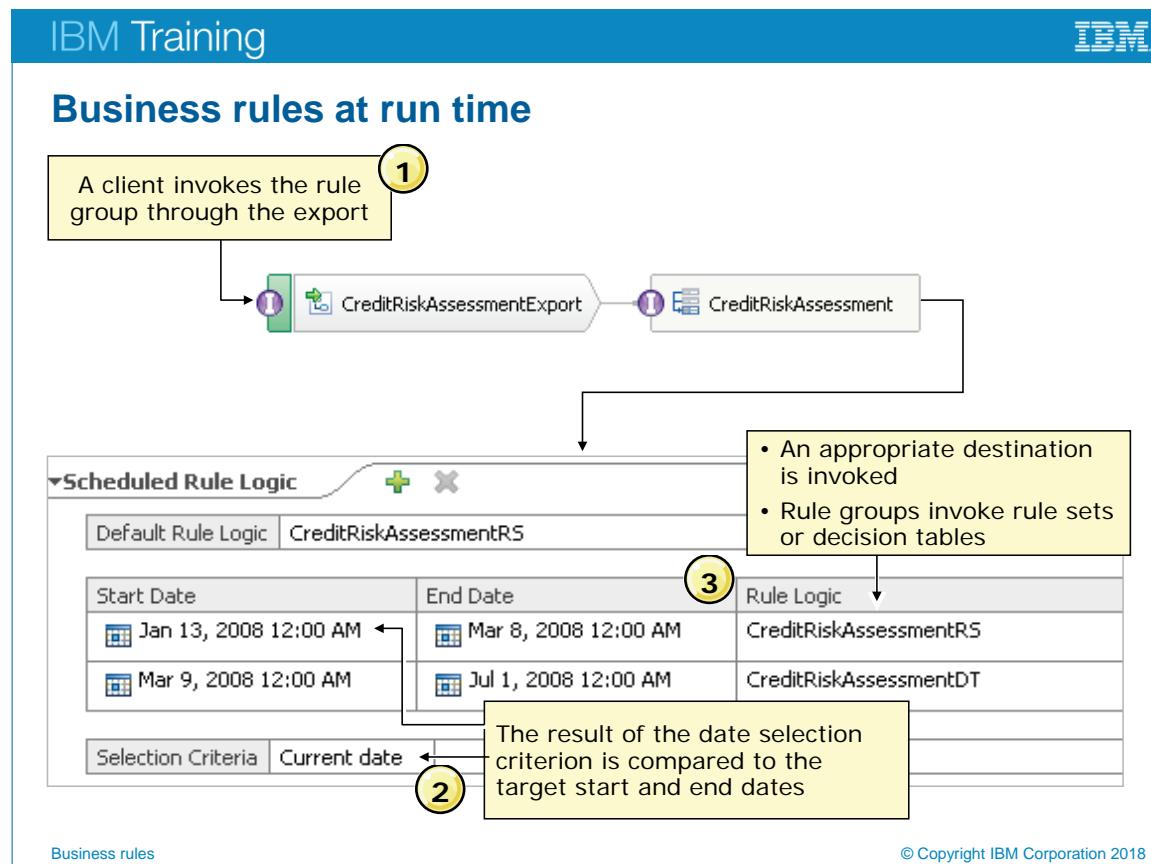
- An activity implementation can contain business rules in IBM Process Designer
 - Called a “decision service”
 - Written in Business Action Language (BAL)
- The following lists differences between rules in IBM Process Designer and IBM Integration Designer:
 - Decision services implement activities or actions only; they do not assign values or run self-contained code
 - BAL is a declarative language, and business rules in IBM Integration Designer support declarative presentations and programmatic rules
 - Rule sets and decision tables are not programmatically selected in IBM Integration Designer

Runtime aspects of rule groups

Business rules

© Copyright IBM Corporation 2018

Runtime aspects of rule groups



Business rules at run time

Rule groups function in the following way at run time:

1. A client (JSP, POJO, SCA component) calls the rule group component interface through the export component.
2. The date that is returned from the date selection criteria is compared to the start and end date criteria of the targets.
 - The date can be “mined” from an attribute in the incoming business object by using XPath or a Java snippet.
 - You can choose to use the current system date.
 - You can use a date that is returned from a Java expression.
3. Based on a comparison of the date selection criteria and the target start and end dates, an appropriate target is invoked.
 - Rule group targets are rule sets and decision tables.
 - If the date criterion falls outside the start and end dates for the rule sets and decision tables, the default destination is used.

More runtime aspects of business rules

- The IBM Process Server runtime handles many business rules
 - However, advanced rule engines like Operational Decision Manager can handle complex rule inference and visualization
- The runtime uses service calls and SCA to support integration with other rule engines
- Business Rules Manager web client and Business Space are used to interact with rules at run time
 - Business Rules Manager is installed with IBM Business Process Manager Advanced
- Business rule information is stored in the IBM Process Server common database (CMNDB) for all supported database types

Business rules

© Copyright IBM Corporation 2018

More runtime aspects of business rules

IBM Process Server handles typical integration rules, but does not handle the other complex rules like inferencing. Complex rules are left to rule engines such as Operational Decision Manager.

The IBM Process Server runtime environment can use the business rules manager web client to edit and update business rules. In addition, Business Space contains a Real-Time Business Configuration widget that can be used to interact with business rules.

Designer and developer roles in business rules

- Exposing rules as templates at run time is based on the separation of responsibilities

Role	Tools used	Tasks
Business analyst (business skills)	Business Rules Manager and Business Space	<ul style="list-style-type: none"> • The business analyst (or architect) designs rules • Analysts adjust rule parameters at run time with business and customer needs in mind
Integration developer (technical skills)	IBM Integration Designer	<ul style="list-style-type: none"> • The developer codes the rules and the rule templates in IBM Integration Designer • SCA components and assembled applications are created by using SOA principles for flexibility and reuse

Business rules

© Copyright IBM Corporation 2018

Designer and developer roles in business rules

The integration developer uses IBM Integration Designer to create or manage the technical details of the business rules. The developer works with the IT architect to implement the business process plan. The business analyst role is more in tune with the changing needs of the business, watching the market, and deciding upon how to provide services (promotional discounts, policy changes, services for a certain age group). Analysts use the business rules manager web client and the Business Space Real-Time Business Configuration widget to change the business rules whenever necessary. The business analyst also works with the developer to update and create business rules for the company.

The Business Rule Manager web tool and the Business Space Real-Time Business Configuration widget are built specifically for the business analyst. They have a manageable subset of full authoring functions that are found in IBM Integration Designer. The only way a business analyst is able to see any rules in the web tools is when the developers build the templates. These templates give the business analyst an easy way to edit business rules at run time by using only a browser.

Business rules manager web client

- The Business Rules Manager client is a browser-based application that can publish changes to business rules or revert to previous rules at run time
- WebSphere Application Server security authenticates the users
- You can change rule group targets, start and end dates, the default rule group, or the decision table

Scheduled Rule Logic
Click button to choose from specifying date, no start/end date, and continuous for automatic end date calculation.

Add Selection Record Sort

Start Date/Time	End Date/Time
Jan 1, 2008, 00:00	Feb 29, 2008, 24:00
Mar 2, 2008, 00:01	Mar 13, 2009, 24:00

Business rules © Copyright IBM Corporation 2018

Business rules manager web client

The business rules manager web client is started through a browser, not the administration console. The business rules manager (BRM) is an enterprise application (a set of JSP pages and servlets), which manages the business rules that are running on the server.

The Business Rule Manager web tool can be accessed by opening a browser and entering: <http://localhost:9080/br/webclient/pages/index.jsp>

You can view all business rules that are exposed through the rule groups that run on the server.

Templates in the business rules manager

- Using the Business Rules Manager web client, you can:
 - Change the rule names and the exposed template parameters
 - Create rules from templates
 - Delete existing rules
 - Change the order of rule execution

Edit Mode: CreditRiskAssessmentRS - Rule Set

Save Cancel Messages:

General Information

Display Name	CreditRiskAssessmentRS	<input checked="" type="checkbox"/> Synchronize with the name
Last Published	Jan 24, 2008 13:00 (Local Time)	Status Original
Description		

Rules

New Rule from Template

You can create a rule from a template, change the execution order, or delete a rule

Name	Display Name	Rule	Description	Action
RiskHIGH	RiskHIGH	If the customer credit score is greater than <input type="text" value="0"/> and less than <input type="text" value="4"/> then the credit risk is <input type="text" value="HIGH"/>	<input type="checkbox"/>	<input type="button" value="Delete"/> <input checked="" type="checkbox"/> Synchronize Name
RiskMED	RiskMED	If the customer credit score is greater than <input type="text" value="3"/> and less than <input type="text" value="8"/> then the credit risk is <input type="text" value="MED"/>	<input type="checkbox"/>	<input type="button" value="Delete"/> <input checked="" type="checkbox"/> Synchronize Name

Business rules © Copyright IBM Corporation 2018

Templates in the business rules manager

The market changes quickly. If web applications do not respond to such changes in a timely manner, business can be lost to competitors. Any minor changes to code would require regression testing, uninstalling the old version of the application, and installing the new application. The business rules manager allows immediate responsiveness and flexibility by allowing a business analyst to change rule parameters at run time.

Exporting and importing rule changes

- Changes that are made to the runtime configuration of rules can be exported, and then imported into IBM Integration Designer
 - Use the command line to export the rules to a compressed file
 - Exported rules are imported into IBM Integration Designer to synchronize runtime and development environments
- Changes in IBM Integration Designer can be imported into the runtime without redeployment

Exporting and importing rule changes

You import rule groups after changing business rules in use by installed applications, and you are ready to bring those changes into another server. You can also use this facility to synchronize your development environment with changes in the production environment. You export rule group components after changing business rules, and you must synchronize your production environment with your development environment. You can use the command line to export and import rule group components. The import function is intended to allow business rule changes to be tested on a test server and then moved over to a production server after testing is complete.

Business rule auditing

- Changes to business rules at run time can be audited
- Audit logging for rules can be configured by using the server command line interface
- You can write changes to `SystemOut.log` or a custom audit log
- Auditing records:
 - Name of person who changes the rule
 - Location where the change request originated
 - Old business rule object
 - New business rule object

Business rule auditing

The business rule objects are the complete business rule set, decision table, or business rule group for the business rule that is replaced and the new version that replaced it. You must examine the logs (the audit output cannot be directed to the Common Event Infrastructure database) to determine the changes that were made, by comparing the old and new business rules.

You can automatically log any changes that are made to business rules. You can configure your server to automatically detect when changes are made to business rules and create an entry in a log file that details the changes. You can choose to have the log entries that are written either to the standard JVM `SystemOut.log` file, or to a custom audit log file of your choice. Depending on how the changes are made, the server where each business rule change is made logs the following information:

- The name of the person who incorporates the change
- The location from where the change request originated
- The old business rule object
- The new business rule object

Public business rules API

- The public business rules API can be used to create custom rule clients similar to the Business Rules Manager
- APIs can be used to:
 - Query business rule groups by name, namespace, or custom properties
 - Modify the business rule group selection table
 - Add and delete rules inside a rule set
 - Add or delete conditions and actions in decision tables
- Use with other APIs to build complete management clients
 - Manage business processes, human tasks, and business rules through a single application

Unit summary

- Define the purpose and business value of using business rules
- Describe the function of a rule group and list the rule group components
- Define the concepts of rule sets and decision tables
- Describe the runtime behavior of a rule group component
- Identify the IBM Process Server administrative capabilities for importing, exporting, and auditing business rule changes in the runtime environment

Checkpoint questions

1. True or False: All rules in a rule set are evaluated in the order that they are written.
2. True or False: You can change the parameters of a business rule at run time without redeploying the application.
3. True or False: A decision table fires only one rule; a rule set might fire several.
4. True or False: Runtime changes to business rules can be exported from IBM Process Server and imported into IBM Integration Designer.

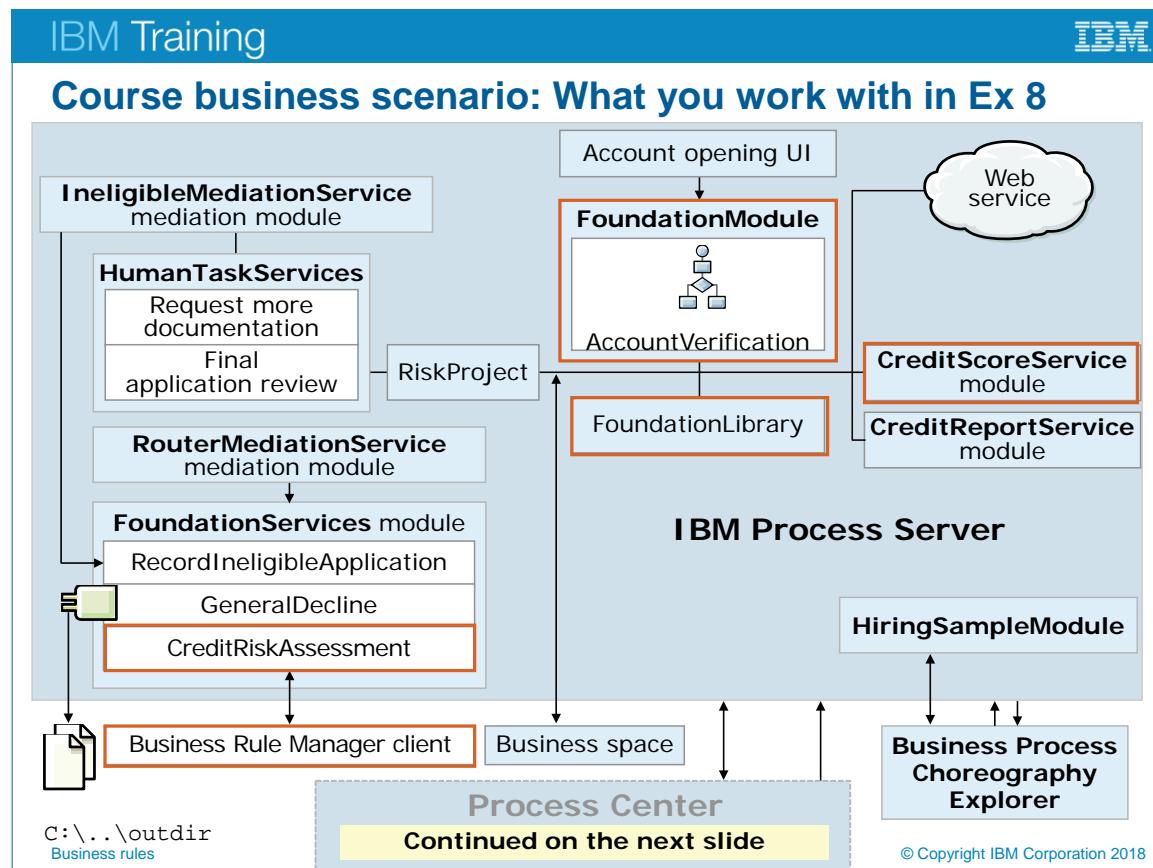
Checkpoint answers

1. True.
2. True: If the developer exposes the rule parameters by using a template.
3. True.
4. True

Exercise 8: Creating business rules

After completing this exercise, you should be able to:

- Create rule sets and decision tables that contain business rules
- Create a rule group component
- Incorporate a rule group component in an assembly diagram
- Test a business rule group in the integration test client
- Use the Business Rule Manager web client to interact with business rules at run time



Course business scenario: What you work with in Ex 8

Components that are required for Exercise 8

Prebuilt components that are imported in the lab:

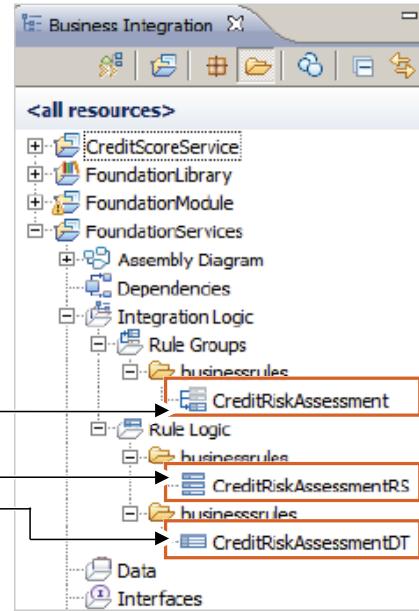
- 1. FoundationModule**
- 2. CreditScoreService**
- 3. FoundationLibrary**
- 4. AccountVerification**

– BPEL process that you completed in Exercise 8

New components that you create in the lab:

- 1. FoundationServices module that contains:**

- **CreditRiskAssessment** rule group
- **MapInputToOutput** rule
- **CreditRiskAssessmentRT** rule set
- **CreditRiskAssessmentDT** decision table



Business rules

© Copyright IBM Corporation 2018

Components that are required for Exercise 8

In this exercise, you create rule sets and decision tables that contain business rules. You create a rule group component, incorporate the rule group component in an assembly diagram, and test it. Finally, you use the Business Rules Manager web client to interact with business rules at run time.

To determine the credit risk, a series of business rules are used to evaluate the credit score that the credit score service returns.

In this exercise, you create the business rules that the credit risk assessment service uses. 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 (short for medium). If the creditScore value is in the range of 8 – 11, then the creditRisk is LOW. Customer applications that are HIGH risk require more documentation and are subject to more review. Customer applications that are MED risk require more review but not more documentation. Customer applications that are LOW risk are automatically approved.

You create a **CreditRiskAssessment** business rule group in the **FoundationServices** module that uses the **CreditRiskAssessment** interface. You also create a **CreditRiskAssessmentDT** decision table in the rule group.

You then create an action rule that is named **MapInputToOutput** that copies the data from the input business object and assigns it to the output business object.

Create rule group in Exercise 8

Scheduled Rule Logic

Default Rule Logic	CreditRiskAssessmentRS
Start Date	Nov 3, 2015 12:00 AM
End Date	Nov 3, 2016 12:00 AM
Rule Logic	CreditRiskAssessmentRS

Rules

Name	RiskHIGH
Template	CreditRiskTemplate
Presentation	If the customer credit score is greater than <input type="text" value="0"/> and less than <input type="text" value="4"/> then the credit risk is HIGH

Name	RiskMED
Template	CreditRiskTemplate
Presentation	If the customer credit score is greater than <input type="text" value="3"/> and less than <input type="text" value="8"/> then the credit risk is MED

Name	RiskLOW
Template	CreditRiskTemplate
Presentation	If the customer credit score is greater than <input type="text" value="7"/> and less than <input type="text" value="12"/> then the credit risk is LOW

© Copyright IBM Corporation 2018

- Test the created application by deploying to a Process Server runtime

Create rule group in Exercise 8

Exercise 8:

Creating business rules

Purpose:

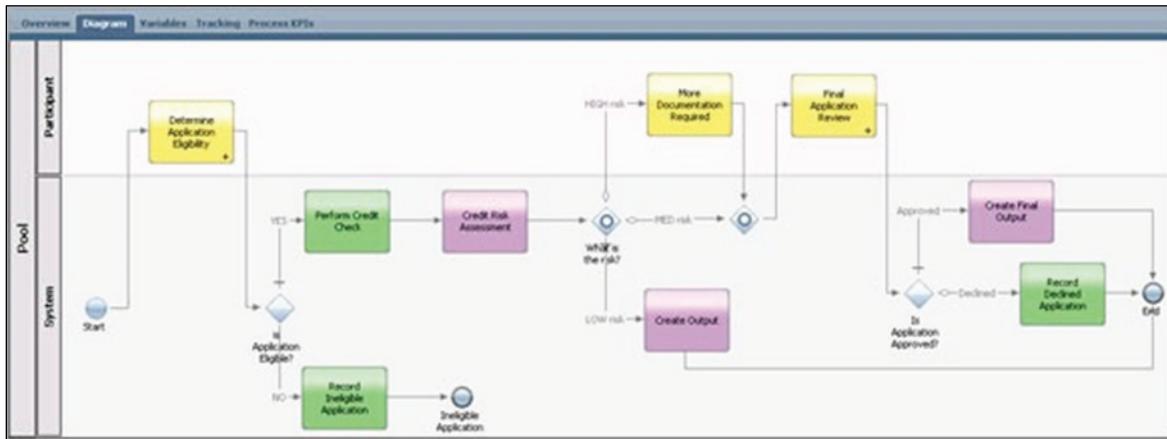
In this exercise, you create rule sets and decision tables that contain business rules. You create a rule group component, incorporate the rule group component in an assembly diagram, and test it. Finally, you use the Business Rules Manager web client to interact with business rules at run time.

A business rule captures and implements business policies and practices. A rule can enforce business policy, incorporate a decision, or infer new data from existing data.

Rules capture decision-making logic in the form of if-then statements. These if-then statements are grouped into rule sets and decision tables. A rule set captures decision-making business logic in the form of a series of if-then statements. A decision table captures multi-conditional decision-making business logic in tabular format. Rule sets and decision tables are incorporated into rule groups. Rule groups are exposed as service components and wired to other components in assembly diagrams.

See the process application model that was created in a previous exercise. In this lab, you implement the Credit Risk Assessment activity that is shown in the model. This activity is implemented as a business rule group.

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.



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 the service that determines a customer's credit risk. To determine the credit risk, a series of business rules are used to evaluate the credit score that the credit score service returns. For educational purposes, you implement the rule logic in both a decision table and a rule set. However, only the rule set is used in your application.

Part 1. Create a rule group component that contains business rules in rule sets and decision tables.

In this portion of the exercise, you create the business rules that the credit risk assessment service uses. 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` (short for medium). If the creditScore value is in the range of 8 – 11, then the creditRisk is `LOW`.

Customer applications that are `HIGH` risk require more documentation and are subject to more review. Customer applications that are `MED` risk require more review but not more documentation. Customer applications that are `LOW` risk are automatically approved.

To create the risk assessment business rules:

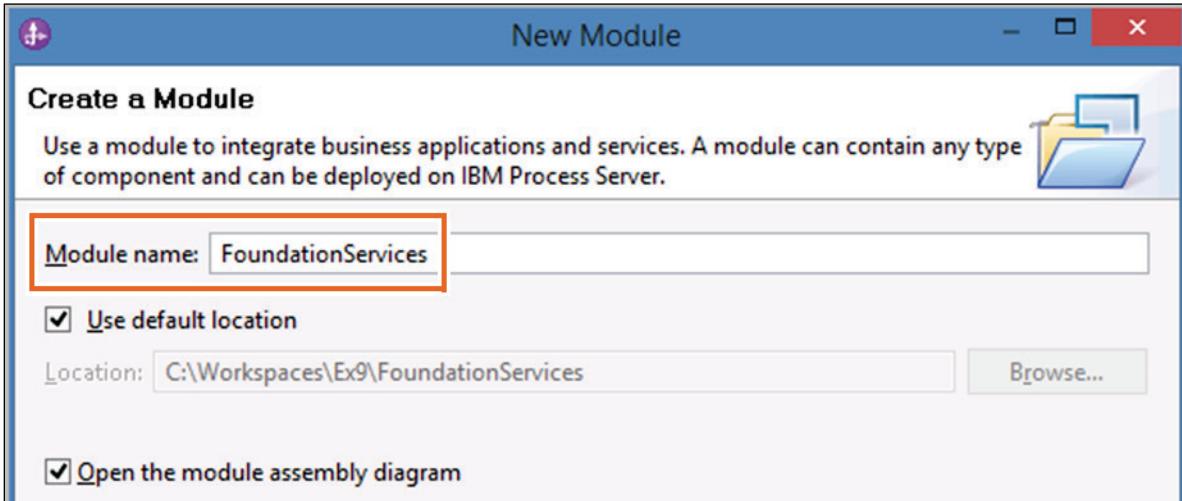
1. Open the Exercise 8 workspace.
 1. On your desktop, open the **Exercise Shortcuts** folder.
 2. Double-click the **Exercise 8** shortcut.

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. Close the **Getting Started** tab.
2. Create a module that is named `FoundationServices` to store the rule group.

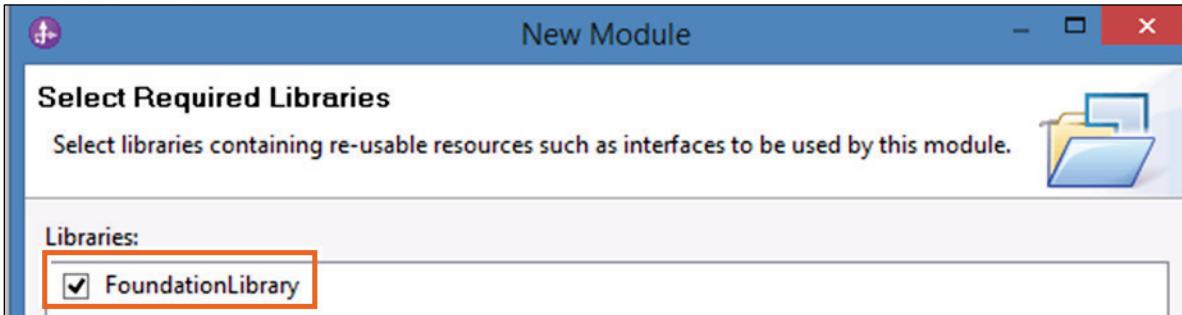
This module is used in later exercises to hold other service implementations that the `AccountVerification` business process invokes.

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

- Type FoundationServices in the **Module name** field.



- Click **Next**.
- In the Select Required Libraries window, select **FoundationLibrary**.

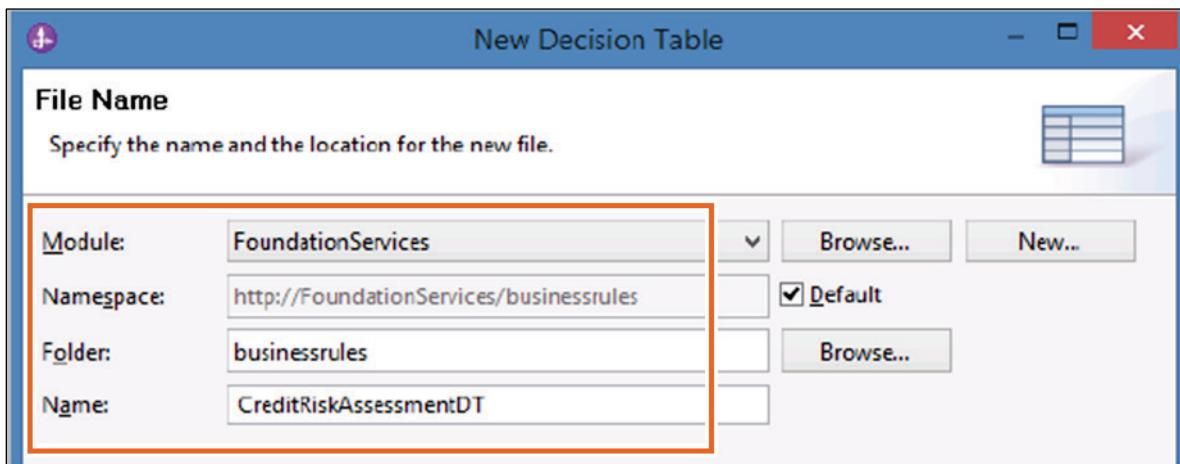


- Click **Finish**.
- Create the decision table for the rule group

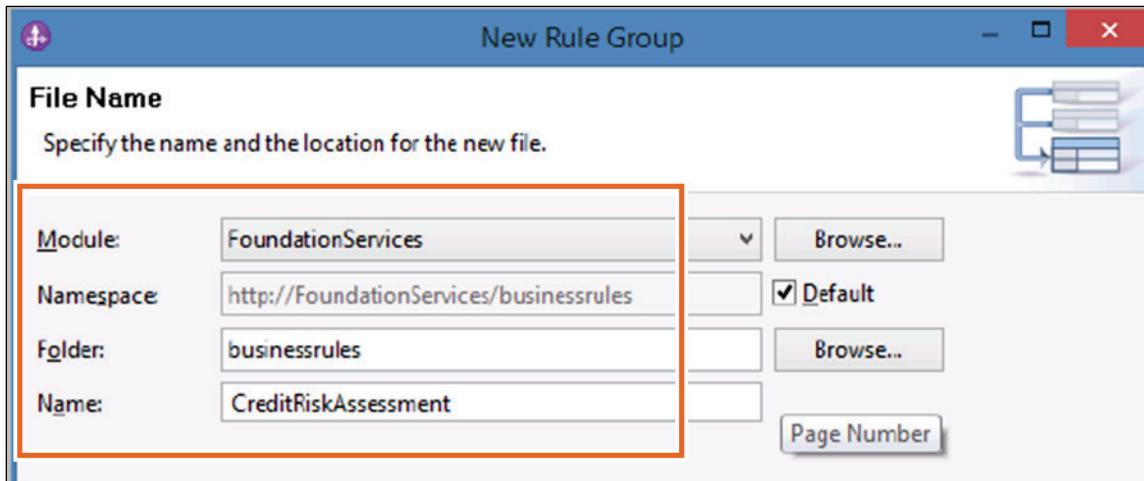
You create a CreditRiskAssessment business rule group in the FoundationServices module in the businessrules/ folder that uses the CreditRiskAssessment interface. Create a CreditRiskAssessmentDT decision table in the rule group.

 - Right-click the FoundationServices module and click New > Rules > Decision Table from the menu.

2. In the **New Decision Table** dialog box, enter the following information.
 - Click **Browse** beside the **Folder** field, click **New Folder**, type **businessrules/** in the **Folder Name** field, and click **OK** twice. When you are returned to the **File Name** window, the slash character is removed from the folder name.
 - Type **CreditRiskAssessmentDT** in the **Name** field.

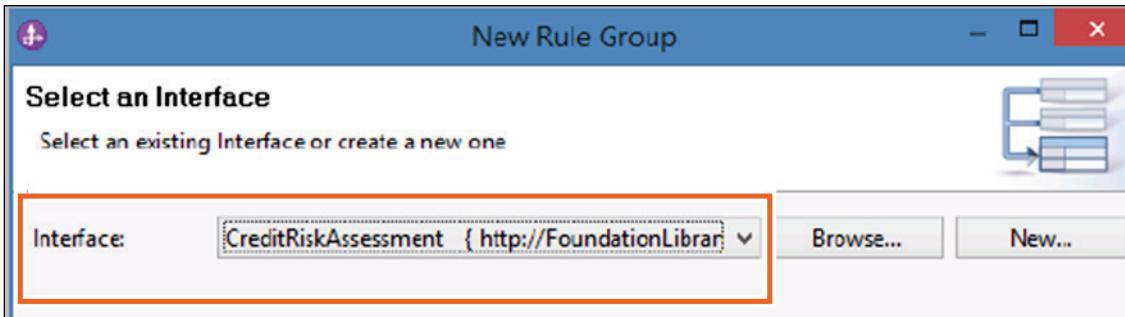


3. Accept the remaining default options and click **Next**.
4. In the “Interface and Operation” window, follow these instructions:
 - Click **New** beside the **Rule Group** field to open the **File Name** window.
 - In the **File Name** window, click **Browse** beside the **Folder** field, select the **businessrules** folder from the list, and click **OK**.
 - When you are returned to the **File Name** window, type **CreditRiskAssessment** in the **Name** field.

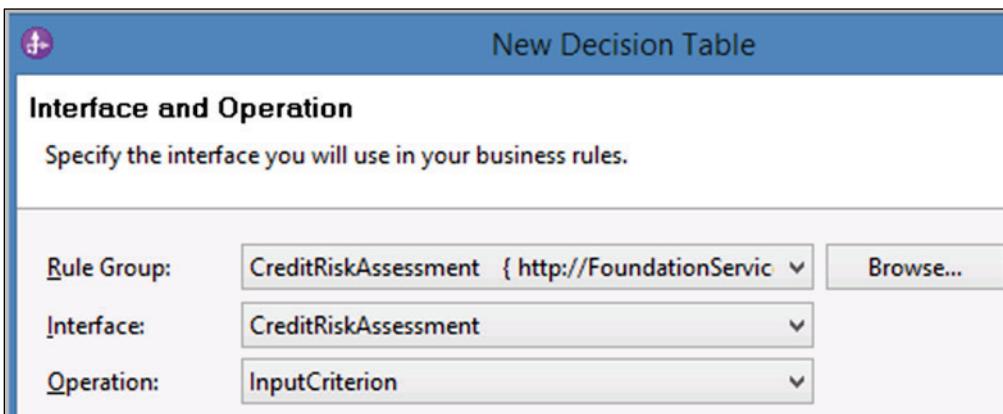


5. Accept the remaining default options and click **Next**.

- In the “Select an Interface” window, select the **CreditRiskAssessment** interface.



- Click **Finish**. When you are returned to the New Decision Table dialog box, on the “Interface and Operation” window, verify that **Rule Group** is set to CreditRiskAssessment, **Interface** is set to CreditRiskAssessment, and **Operation** is set to InputCriterion.



- Click **Finish** to complete the decision table wizard.
- Create an initialization action rule that is named `MapInputToOutput` 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:

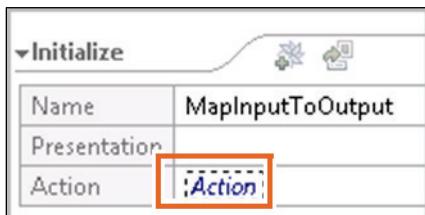
```
Output = copyBO( Input )
```

- In the **Initialize** section of the decision table editor, click the **Add an Initialization Action Rule** icon.

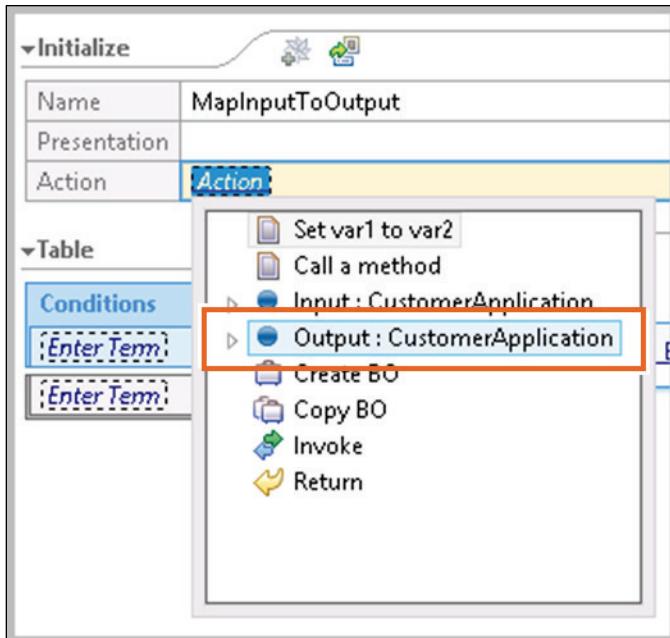


- Change the action rule **Name** from `Rule1` to: `MapInputToOutput`

- In the **Action** field, click the **Action** link to create an expression with the expression builder.

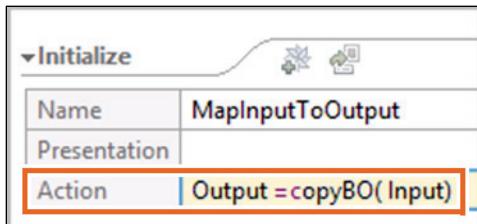


- Select **Output : CustomerApplication** from the list.



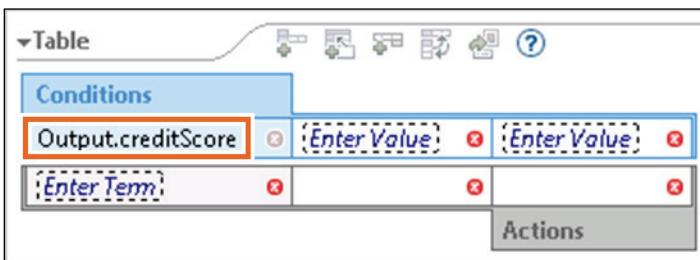
- Select the = operator (the first item in the list).
- Select **Copy BO**.
- Select **Input: CustomerApplication**.
- Click the **Action** field to close the expression builder.

Your completed action rule resembles the following figure:

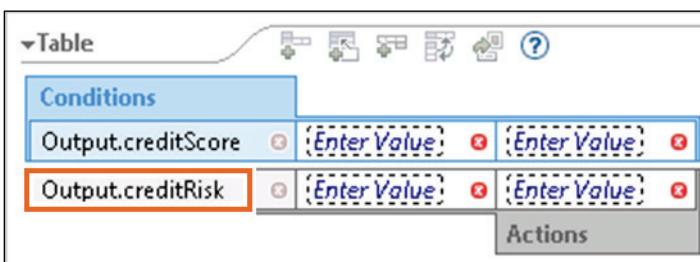


Note: You can type the entire expression manually instead of using the expression builder, but the expression builder helps eliminate the possibility of typographical errors.

9. Save your changes. You can ignore any errors in the **Problems** view.
4. Use the following business rules to implement the decision table.
 - 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 **Table** section of the editor, click the **Enter Term** link in the first row of the **Conditions** column.
2. Using the expression builder, expand **Output : CustomerApplication** and select **creditScore** (click the word **Conditions** to close the expression builder). Alternatively, type: `Output.creditScore`

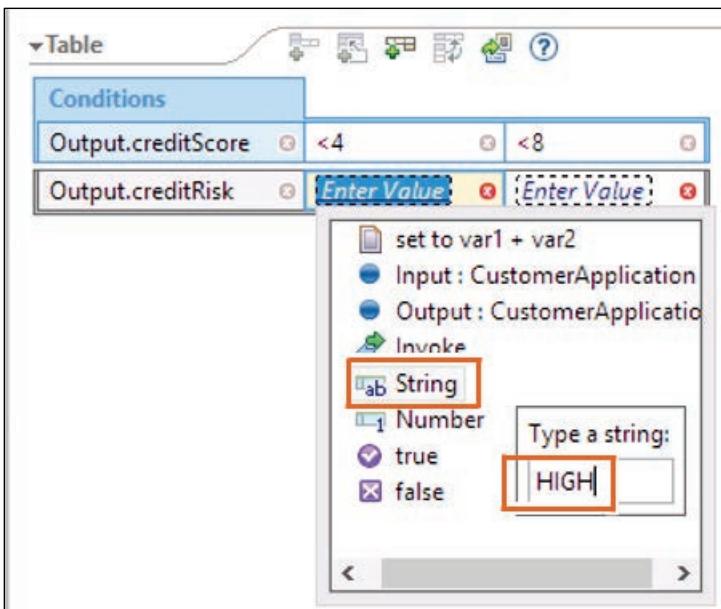


3. Click the **Enter Term** link in the second row of the **Conditions** column.
4. Using the expression builder, expand **Output : CustomerApplication** and select **creditRisk**. Alternatively, type: `Output.creditRisk`



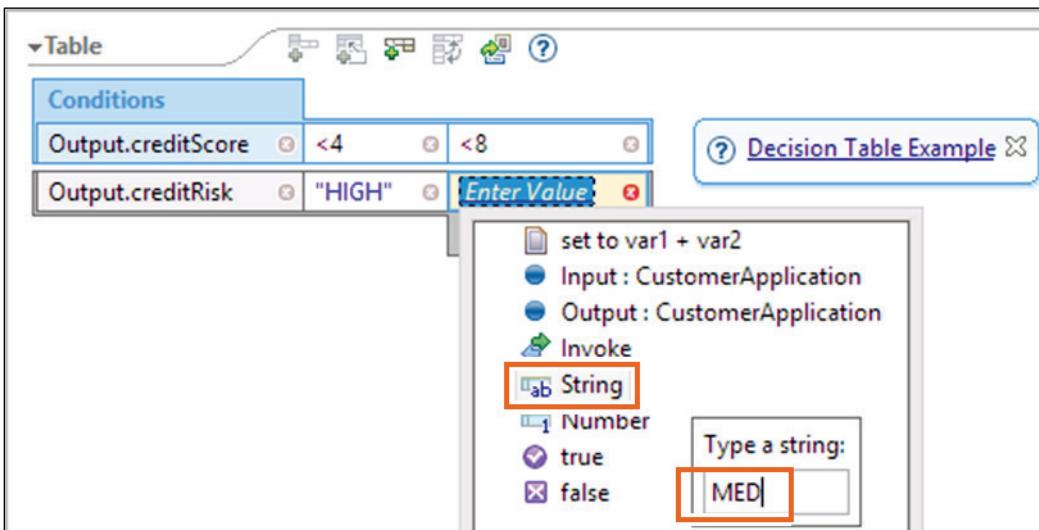
5. Click the **Enter Value** link in the first row, second column, and type the expression: `<4`
6. Click the **Enter Value** link in the first row, third column, and type the expression: `<8`
7. Click the **Enter Value** link in the second row, second column.

8. Select **String**, type **HIGH** in the **Type a string** field, and press Enter. This rule sets `Output.creditRisk` to **HIGH** when the credit score is less than 4.



Note: Though you entered **HIGH** in the **Type a string** field without quotation marks, the value is displayed in the decision table *with* quotation marks, which is normal for a string value.

9. Click the **Enter Value** link in the second row, third column.
10. Select **String**, type **MED** in the **Type a string** field, and press Enter. This rule sets `Output.creditRisk` to **MED** (short for medium) when the credit score is in the range of 4 – 7.



11. Click the **Add a New Condition Value** icon to add a column to the table. If the icon is not active, select the `<8` table cell first.



12. Click the **Enter Value** link in the first row, fourth column, and type the expression: <12

13. Click the **Enter Value** link in the second row, fourth column.

14. Select **String**, type **LOW** in the **Type a string** field, and press Enter.

This rule sets `Output.creditRisk` to `LOW` when the credit score is in the range of 8 – 11.

The screenshot shows a decision table with two rows and four columns under the 'Conditions' tab. The first row has conditions <4, <8, and <12, with values 'HIGH', 'MED', and 'Enter Value' respectively. The second row has conditions <4, <8, and <12, with values 'HIGH', 'MED', and 'LOW'. A context menu is open over the 'Enter Value' cell in the second row, showing options like 'set to var1 + var2', 'Input: CustomerApplication', 'Output: CustomerApplication', 'Invoke', and 'String'. A sub-menu for 'String' is open, showing 'Type a string:' with the value 'LOW' entered. The 'LOW' entry is highlighted with a red box.

15. Place your cursor in the <12 cell (first row, fourth column), right-click the cell, and select **Add Condition Otherwise** from the menu.

16. Click the **Enter Value** link in the **Otherwise** column.

17. Select **String**, type **HIGH** in the **Type a string** field, and press Enter. This rule sets `Output.creditRisk` to `HIGH` if the credit score is outside the normal range of 1 to 11.

The screenshot shows a decision table with two rows and four columns under the 'Conditions' tab. The first row has conditions <4, <8, and <12, with values 'HIGH', 'MED', and 'LOW'. The second row has conditions <4, <8, and 'Otherwise', with values 'HIGH', 'MED', and 'Enter Value'. A context menu is open over the 'Enter Value' cell in the second row, showing options like 'set to var1 + var2', 'Input: CustomerApplication', 'Output: CustomerApplication', 'Invoke', and 'String'. A sub-menu for 'String' is open, showing 'Type a string:' with the value 'HIGH' entered. The 'HIGH' entry is highlighted with a red box.

18. The completed **CreditRiskAssessmentDT** decision table resembles the following image.

The screenshot shows a decision table with two rows of conditions and four columns of actions. The first row has four conditions: <4, <8, <12, and Otherwise. The second row has four actions: "HIGH", "MED", "LOW", and "HIGH".

Conditions				
Output.creditScore	<4	<8	<12	Otherwise
Output.creditRisk	"HIGH"	"MED"	"LOW"	"HIGH"
Actions				

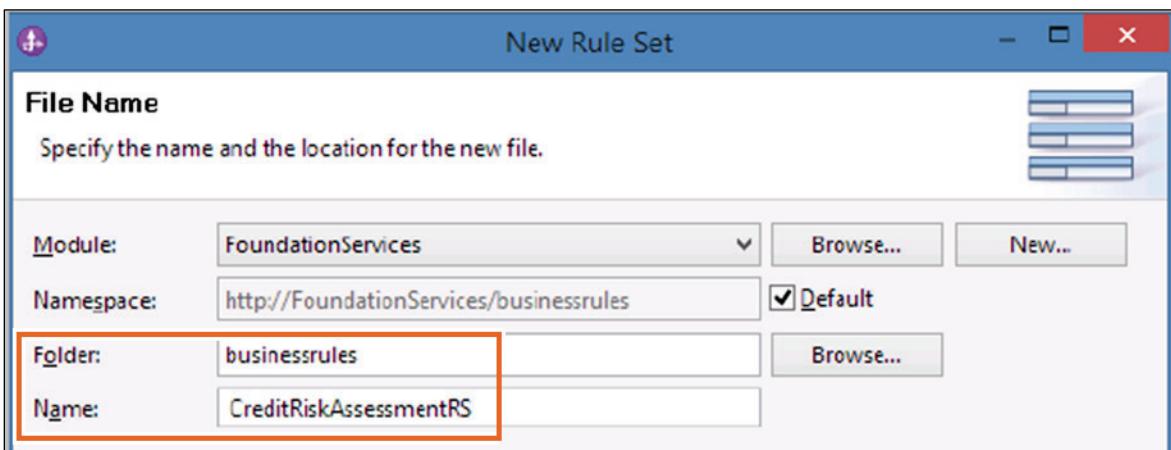
19. Save your changes and close the decision table editor by closing the tab. Continue to ignore any errors in the **Problems** view.

5. Create the rule set for the rule group

You create a rule set that is named `CreditRiskAssessmentRS` in the `FoundationServices` module, in the `businessrules/` folder, that uses the `CreditRiskAssessment` interface.

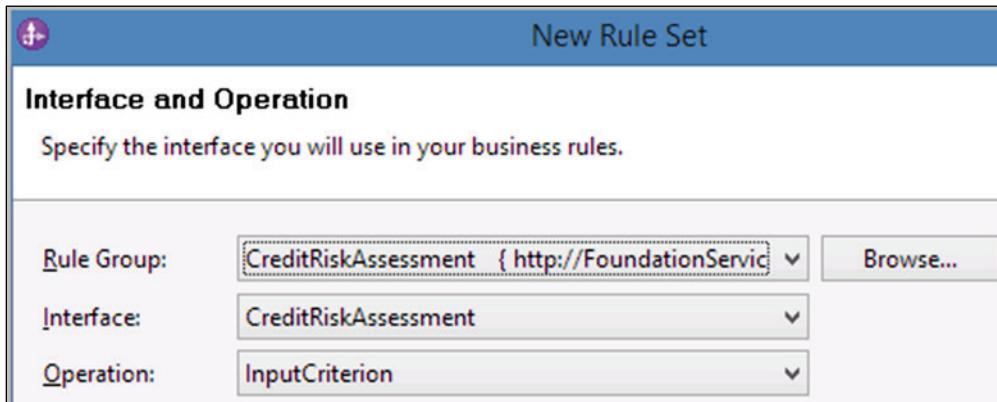
You are creating both a rule set and a decision table for educational purposes. Only the rule set is used in your application.

1. Expand `FoundationServices > Integration Logic`.
2. Right-click **Rule Logic** and click **New > Rule Set** from the menu.
3. In the File Name window, follow these instructions:
 - Click **Browse** beside the **Folder** field, select the **businessrules** folder, and click **OK**.
 - Type `CreditRiskAssessmentRS` in the **Name** field.



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

5. In the “Interface and Operation” window, enter the following information.
- In the **Rule Group** field, select **CreditRiskAssessment**.
 - Verify that the **Interface** is set to CreditRiskAssessment and the **Operation** is set to InputCriterion.



6. Click **Finish**.
6. Create an if-then rule template that is named CreditRiskTemplate with three parameters: lowerLimit (of type int), upperLimit (of type int), and decision (of type string).

A rule set template defines the implementation and parameters for an if-then or action rule. When defined, the template can be used to create instances of the same rule by using different parameters.

The template logic is:

If all of the following are true

- Output.creditScore > lowerLimit
- Output.creditScore < upperLimit

Then Output.creditRisk = decision

The presentation of the template is:

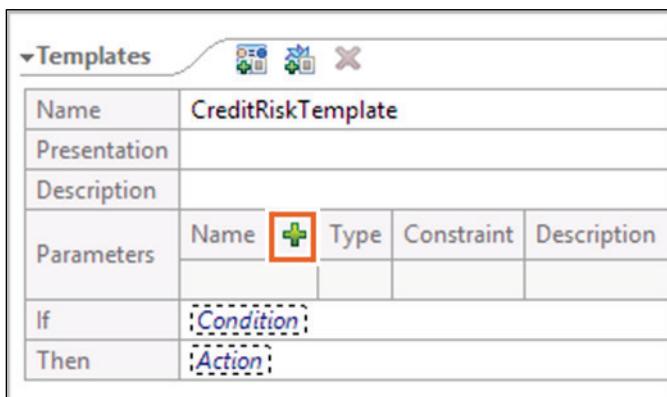
If the customer credit score is greater than lowerLimit and less than upperLimit then the credit risk is decision

1. In the **Templates** section of the rule set editor, click the **Add If-Then Template** icon.

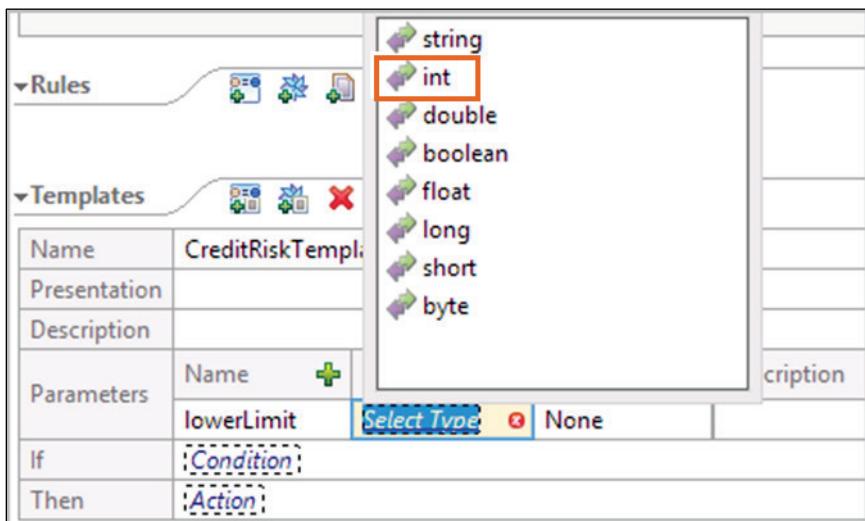


2. In the **Name** field, replace Template 1 with: CreditRiskTemplate

- In the **Parameters** row, click the plus sign (+) icon beside **Name** to add a parameter.



- In the **Name** column, change the parameter name from `param1` to: `lowerLimit`
- In the **Type** column, click the **Select Type** link and select **int**.



- In the **Parameters** row, click the plus sign (+) icon beside **Name** to add a parameter.
- In the **Name** column, change the parameter name from `param1` to: `upperLimit`
- In the **Type** column, click the **Select Type** link and select **int**.

	Name	+	Type	Constraint
Parameters	lowerLimit	+	int	None
	upperLimit	+	int	None

- In the **Parameters** row, click the plus sign (+) icon beside **Name** to add a parameter.
- In the **Name** column, change the parameter name from `param1` to: `decision`

11. In the **Type** column, click the **Select Type** link and select **string**.

	Name	Type	Constraint
Parameters	lowerLimit	int	None
	upperLimit	int	None
	decision	string	None

12. In the **If** row, click the **Condition** link and select **All of the following are true** from the list.

Templates	
Name	var1 is equal to var2
Presentation	var1 is greater than var2
Description	All of the following are true
Parameters	Any of the following is true Input : CustomerApplication Output : CustomerApplication lowerLimit : int upperLimit : int decision : string
If	Condition
Then	Action

13. Under “all of the following are true,” click the first **Condition** link.
 14. In the expression builder list, expand **Output : CustomerApplication** and select **creditScore**.
 15. Select the greater than (**>**) operator.
 16. Select **lowerLimit : int**. The completed condition resembles the following image:

If	all of the following are true • Output.creditScore > lowerLimit • Condition
----	---

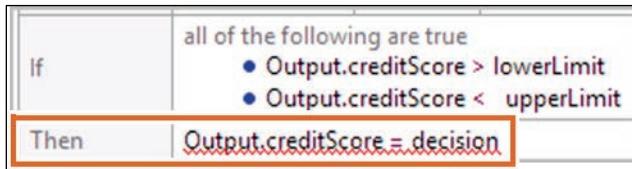
17. Click the second **Condition** link under “all of the following are true.”
 18. Expand **Output : CustomerApplication** and select **creditScore** from the list.
 19. Select the less than (**<**) operator.
 20. Select **upperLimit : int**. The completed “If” section of the template resembles the following image:

If	all of the following are true • Output.creditScore > lowerLimit • Output.creditScore < upperLimit
----	---

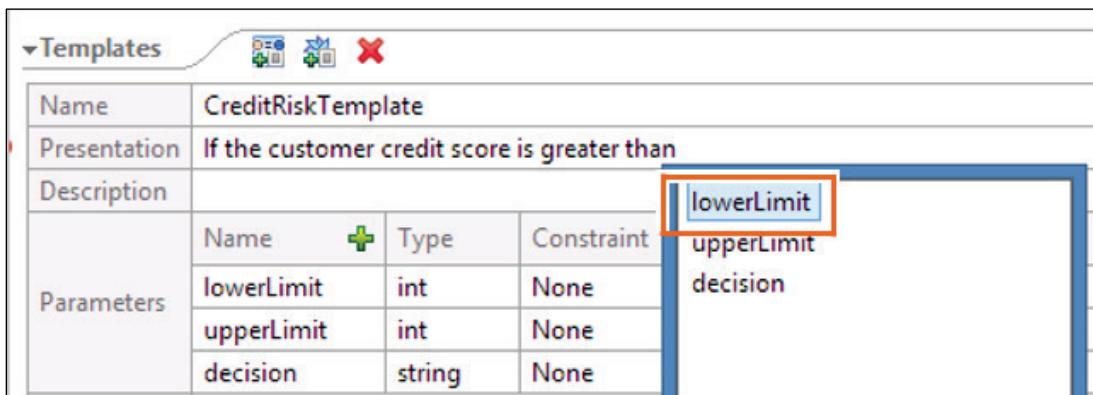
21. In the **Then** row, click the **Action** link.

22. Expand **Output : CustomerApplication** and select **creditRisk** from the list.
23. Select the equal sign (=) operator.
24. Select **decision : string**.

The completed “Then” portion of the template resembles the following image:



25. In the **Presentation** field, type: If the customer credit score is greater than (insert a space at the end of the text).
26. Click the **Presentation** field after the space at the end of the word than to reveal the expression builder.
27. Select **lowerLimit** from the list.



28. After **lowerLimit**, type a space followed by the text and less than followed by another space.
29. Click the **Presentation** field after the space at the end of the word than to reveal the expression builder.
30. Select **upperLimit** from the list.
31. After **upperLimit**, type a space followed by the text then the credit risk is followed by another space.
32. Click the **Presentation** field after the space at the end of the word is to reveal the expression builder.
33. Select **decision** from the list.