

Hello World: WebSphere Enterprise Service Bus

Develop message flow for protocol transformation

Skill Level: Intermediate

[Abelard Chow \(abchow@ca.ibm.com\)](mailto:abchow@ca.ibm.com)

Advisory Software Developer
IBM

[Bill Zhu \(billzhu@ca.ibm.com\)](mailto:billzhu@ca.ibm.com)

Software Developer
IBM

07 Nov 2006

Learn how to build message flow for protocol transformation. This tutorial demonstrates protocol transformation characteristics of WebSphere® Enterprise Service Bus. This is the fifth tutorial in the "Hello, World" series, which provides high-level overviews of various IBM® software products. WebSphere Enterprise Service Bus is designed to meet the connectivity and integration needs of Web Services applications and data. Customers who are interested improving their competitive edge with business transformation should consider an Enterprise Service Bus solution leveraging one of IBM's ESB products, such as WebSphere Enterprise Service Bus.

Section 1. Before you start

About this series

This series is for novices who want high-level overviews of various IBM software products. The modules are designed to introduce the products and draw your interest for further exploration. The exercises only cover the basic concepts, but are

enough to get you started.

About this tutorial

This tutorial provides introductory product information, basic concepts of ESB, and a hands-on exercise that demonstrates how to build a mediation module for protocol transformation using Service Component Architecture and WebSphere Integration Developer.

Objectives

After completing this tutorial, you should understand:

- The basic concepts of ESB
- How to build message flow for protocol transformation using SCA
- How to use Integration Test Client in WebSphere Integration Developer
- How to generate a Web Service proxy and sample JSP pages

Prerequisites

This tutorial is designed for developers with minimum or no knowledge of WebSphere Enterprise Service Bus.


System requirements

Before starting this tutorial, make sure WebSphere Integration Developer V6.0.1.1 is installed. Learn about this product at <http://www-306.ibm.com/software/integration/wid/>.

To view the demos included in this tutorial, enable JavaScript in your browser and install Macromedia Flash Player 6 or higher. Download the latest Flash Player at <http://www.macromedia.com/go/getflashplayer/>.

Animated demos

If this is your first encounter with a developerWorks tutorial that includes demos, here are a few helpful hints:

- Demos are an optional way to see the same steps described in the tutorial. To see an animated demo, click the  Show me link. The demo opens in a new browser window.
 - Each demo contains a navigation bar at the bottom of the screen. Use the navigation bar to pause, exit, rewind, or fast forward portions of the demo.
 - The demos are 800 x 600 pixels. If this is the maximum resolution of your screen, or if your resolution is lower than this, scrolling is necessary to see some areas of the demo.
-

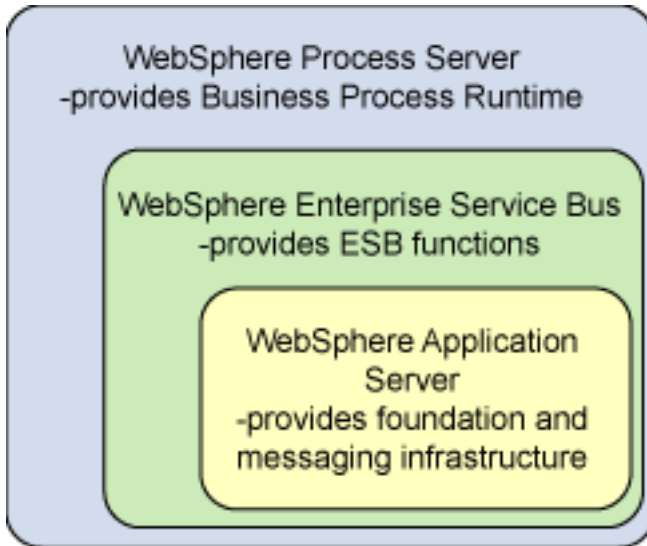
Section 2. Introduction

What is WebSphere Enterprise Service Bus?

WebSphere Enterprise Service Bus is a new IBM product that entered the market in late 2005. It is designed to meet the connectivity and integration needs of Web Service applications and data. WebSphere Integration Developer is the development platform for WebSphere Enterprise Service Bus. It provides a flow editor, which allows developers to build mediation modules that deploy on WebSphere Enterprise Service Bus, as part of the Service Component Architecture (SCA). WebSphere Enterprise Service Bus, together with WebSphere Integration Developer and Service Component Architecture, allows developers to build, to deploy and to manage Enterprise Service Bus solutions.

The following diagram illustrates how WebSphere Enterprise Service Bus relates to other WebSphere server editions:

Figure 1. WebSphere Enterprise Service Bus and other WebSphere server editions



Customers can realize the following benefits when using an Enterprise Service Bus business transformation solution, such as WebSphere Enterprise Service Bus:

- Reduce development and maintain costs by moving to a service base architecture.
- Improve business responsiveness by getting products to the market faster.
- Increase customer satisfaction by improving product quality and delivery speed.
- Maximize return on assets by reusing existing business process.
- Bring new products to the existing customer with cross selling.
- Outwit the competitors by building differentiated applications.

WebSphere Enterprise Service Bus key concepts

Key concepts	Definition
Service requester and service provider	Two types of application connect to an Enterprise Service Bus -- service requesters and service providers. Service requesters are applications that require service from another application on an ESB. Service providers are applications that provide service to other applications on an ESB.
Mediation module	A mediation module is an SCA artifact that processes and

	mediates message flows between service requesters and service providers.
Mediation flow	Mediation flow contains high-level mediation logic. It describes how messages flow between service requesters and service providers.
Mediation primitive	Mediation primitives are nodes connected within mediation flows. They contain programming logic that performs specific business or system functions.
Service message object	Service message objects are enhanced service data objects. It is an abstract layer for message processing between service requesters and service providers.
Application server	Mediation modules are deployed on a WebSphere Enterprise Service Bus-enabled application server.

WebSphere Enterprise Service Bus key features

WebSphere Enterprise Service Bus provides the following features in addition to those provided by WebSphere Application Server:

- Routes messages between services
- Converts transportation protocol between service requesters and service providers
- Transforms message formation between service requesters and service providers
- Handles business events

Section 3. What is Enterprise Service Bus?

The concept of Enterprise Service Bus was first introduced in 2002. Since then, it has been widely adapted in the software industry. Enterprise Service Bus provides an integration platform that combines Message Oriented Middleware (MOM)

integration strategy, Web Services, Message Transformation, and Intelligence Message Routing into an event-driven Service Oriented Architecture (SOA).

Message Oriented Middleware

Message Oriented Middleware allows applications running on different servers to connect in a loosely coupled, asynchronous fashion through messages communication. It is a flexible approach to architect distributed systems.

Web Services

It is widely acknowledged that Web Services is the cornerstone of Service Oriented Architecture. Web Services provides an architecture and standard for the industry to implement applications as reusable services. These reusable services are accessible by any potential service requester.

Message Transformation

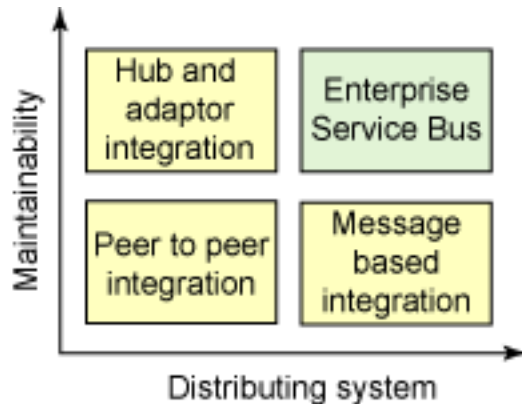
The ability of data mapping and data transformation is essential in any application integration strategy. An Enterprise Service Bus must be able to transform the message sent from a service requester into a format receivable by the service provider, and vice versa.

Intelligence Message Routing

Message Routing, an essential ability in Enterprise Service Bus, allows service requesters truly decoupled from the service providers. Messages sent from a service requester may be routed to one or more service providers based on the routing logic.

Enterprise Service Bus is an SOA application integration architecture; however, how does it differ from conventional integration approaches? The following diagram illustrates how conventional integration approaches compare to Enterprise Service Bus:

Figure 2. Conventional integration approaches vs. ESB



Conventionally, applications can be integrated in certain ways: Peer-to-peer integration, Hub and Adaptor integration and Message-based integration. Peer-to-peer integration resulted from an ad hoc approach. It is infamous for non-scaling and difficult to maintain. The Hub and Adaptor integration approach connects applications to hubs using various adaptors. It is more maintainable than Peer-to-peer, but is not a good solution for distributed systems. In Message-based integration, applications are loosely coupled through message exchange. This approach makes a system more distributable; however, existing applications that do not support messaging require massive modification to be integrated.

Enterprise Service Bus combines the benefit of Hub and Adaptor integration and Message-based integration to provide maximum maintainability and distributing ability.

Section 4. How does WebSphere Enterprise Service Bus fit into the IBM Solution?

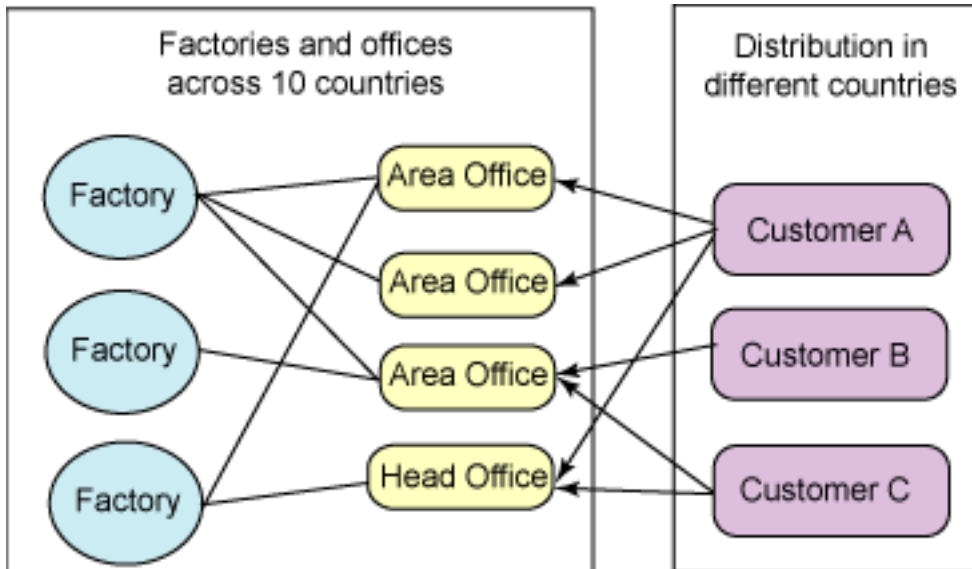
Three out of four chief executive officers today believe the ability to respond rapidly to the business environment is essential to their company's success. However, only 10 percent of them believe their company has the ability to do so. The key to improve responsiveness is an application and service integration infrastructure that enables development of SOA and incremental business transformation. As a result, information can be delivered to other departments and business partners as quickly, economically, and flexibly as possible. Their companies will have the ability to grow as needed.

In respond to the needs of the business market, IBM released WebSphere Enterprise Service Bus, which is one of the IBM ESB solutions. It provides customers a complete SOA solution for application and service integration, from managing Web Services connectivity, interaction, and service hosting to service

mediation.

The following diagrams illustrate a scenario where ESB can be applied to resolve business issues. A manufacturing enterprise has many offices and factories across 10 countries. It supplies many distributors worldwide.

Figure 3. Existing system of a manufacturing enterprise

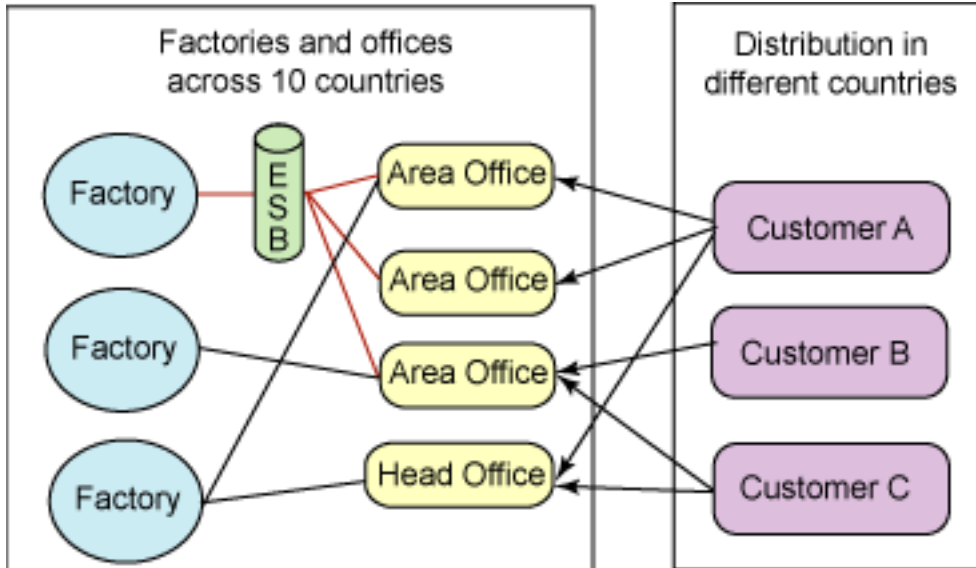


Under the current integration infrastructure, this company may experience the following problems:

- Production at a factory exceeds, or fails to meet, demand.
- Synchronization of orders from different offices requires heavy administration.
- Inadequate information sharing between offices and factories incurs extra shipment costs.

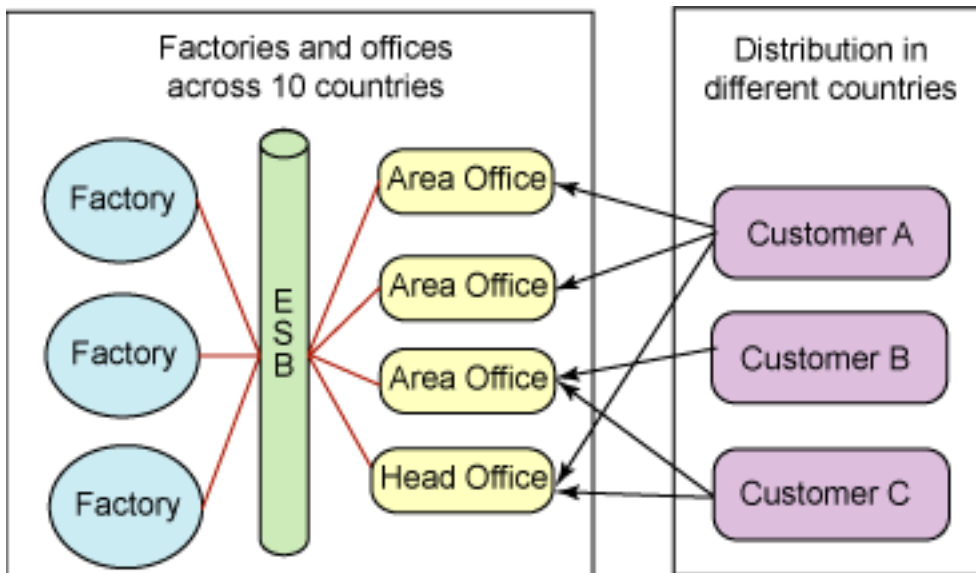
An ESB solution, using IBM WebSphere Enterprise Service Bus, can be introduced into this enterprise to incrementally transform the existing business and solve this company's issues step by step. As a first step, ESB can be used to transform part of the organization's system into SOA infrastructure. By completing transformation as illustrated in the following diagram, orders to the first factory are synchronized across the connected offices. Any of the offices can obtain real-time status of the factory.

Figure 4. Introduce ESB to existing system



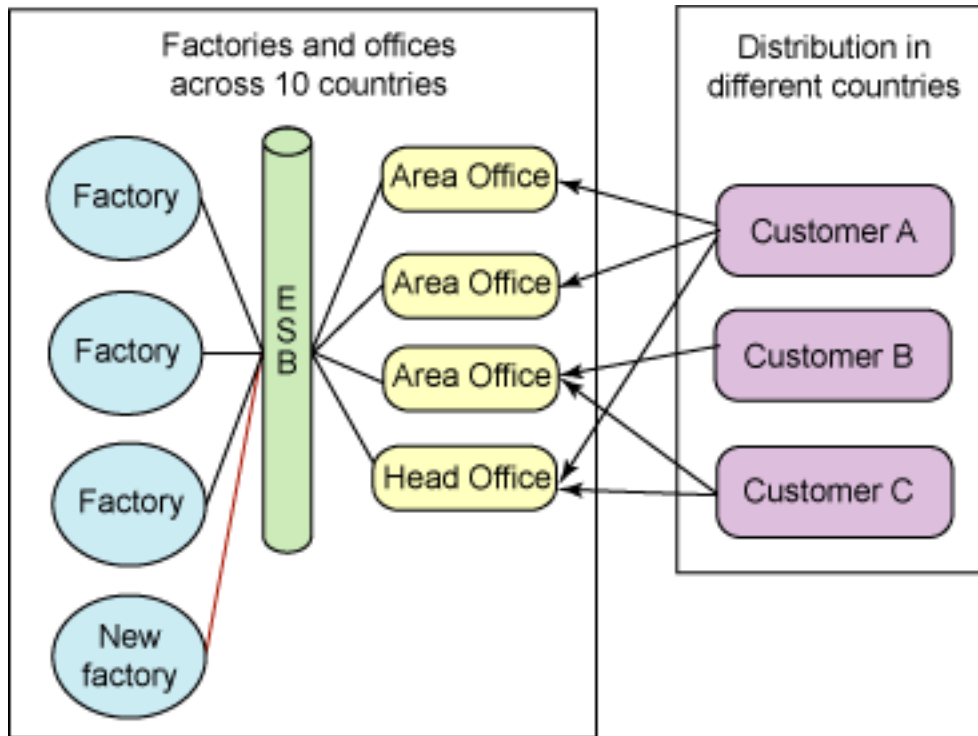
Later on, further steps can be taken to eventually transform the whole system and business. All of the existing issues are resolved with the new ESB solution. Each incremental step this organization takes brings immediate benefit into the enterprise.

Figure 5. Transform the whole system incrementally



Once the whole enterprise is completed transformed, it will have the ability to adapt rapidly to any changes in the business environment. Suppose customers increase their orders. The enterprise can quickly evaluate if the new order can be absorbed by the existing factories. Is the best way to handle the extra orders to open up a new factory, order from other suppliers, or ship products across the world? Because the system is already SOA enabled, business alternatives are easily integrated into the enterprise.

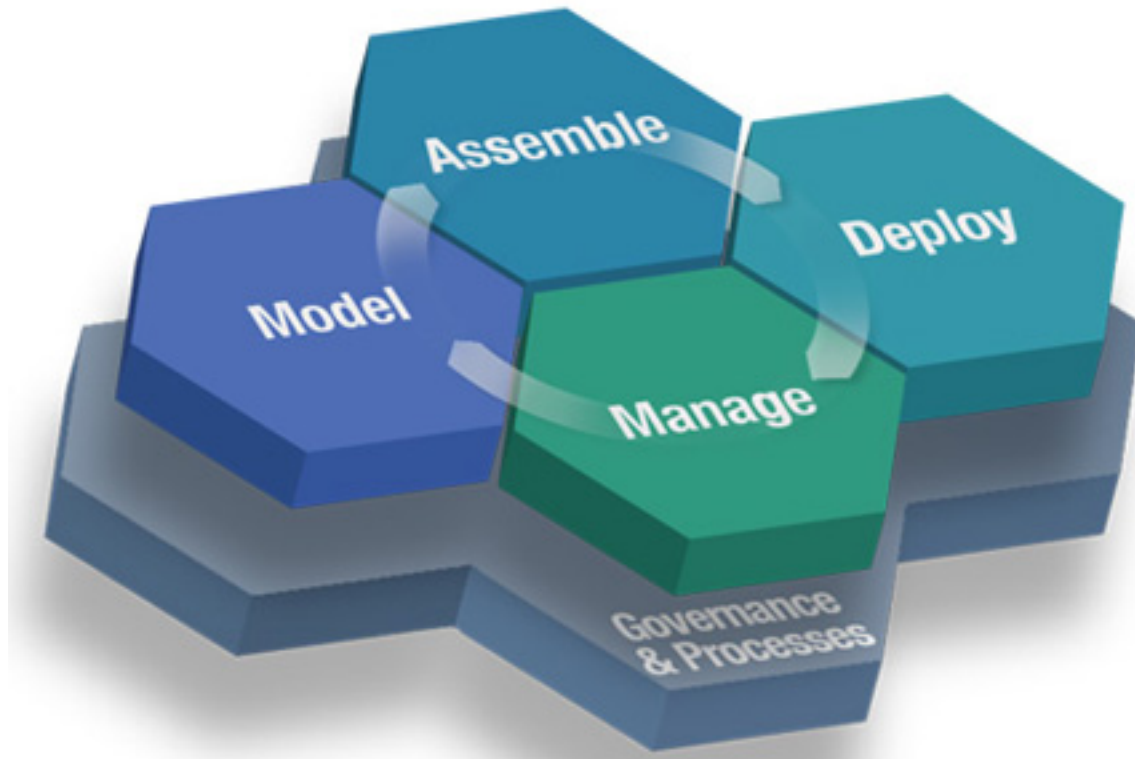
Figure 6. Integrate new alternative into a SOA system



Section 5. How does WebSphere Enterprise Service Bus fit into SOA?

To help customers build and manage business processes, IBM introduced the Service Oriented Architecture (SOA) Foundation. The SOA Foundation is an integrated, open set of software, best practices, and patterns that provide what customer needs to get started with the SOA life cycle. Following is a diagram of SOA life cycle:

Figure 7. SOA life cycle



WebSphere Enterprise Service Bus is one of the software products in the SOA Foundation that support the deploy phase of SOA life cycle. WebSphere Integration Developer is one of the software products that supports the assemble phase.

Section 6. Service Component Architecture and WebSphere Enterprise Service Bus

Application integration infrastructure such as WebSphere Enterprise Service Bus is very powerful; however, powerful infrastructure does not usually imply easy to use. Developers may find themselves spending more time to learn or use an infrastructure rather than focus on developing business logic. To help companies reduce learning costs and to help developers pick up ESB development faster, IBM established Service Component Architecture (SCA).

Service Component Architecture is one way of implementing SOA architecture. It provides a technology-independent model that specifies interface, reference, and implementation. Such a model is known as a SCA component. The advantage of using SCA is that infrastructure logic is separated from business logic, which enables developers to focus on resolving the business side of technical issues.

Currently, SCA components can be implemented in the following ways with WebSphere Integration Developer:


- Java
 - BPEL
 - State Machine
 - Business Rule
 - Human Task
 - Selector
 - Mediation Flow
-

Section 7. Protocol transformation

This exercise lets you practice using WebSphere Integration Developer and Service Component Architecture (SCA) to implement a WebSphere Enterprise Service Bus message flow that transforms a SOAP/HTTP request from the service requester to SOAP/JMS for the use of service provider.

As stated in [Prerequisites](#), you need to have WebSphere Integration Developer V6.0.1.1 and WebSphere Enterprise Service Bus already installed to complete this exercise. To create a mediation module, complete the following steps:

Would you like to see these steps demonstrated for you?

 [Show me](#)

Starting WebSphere Integration Developer

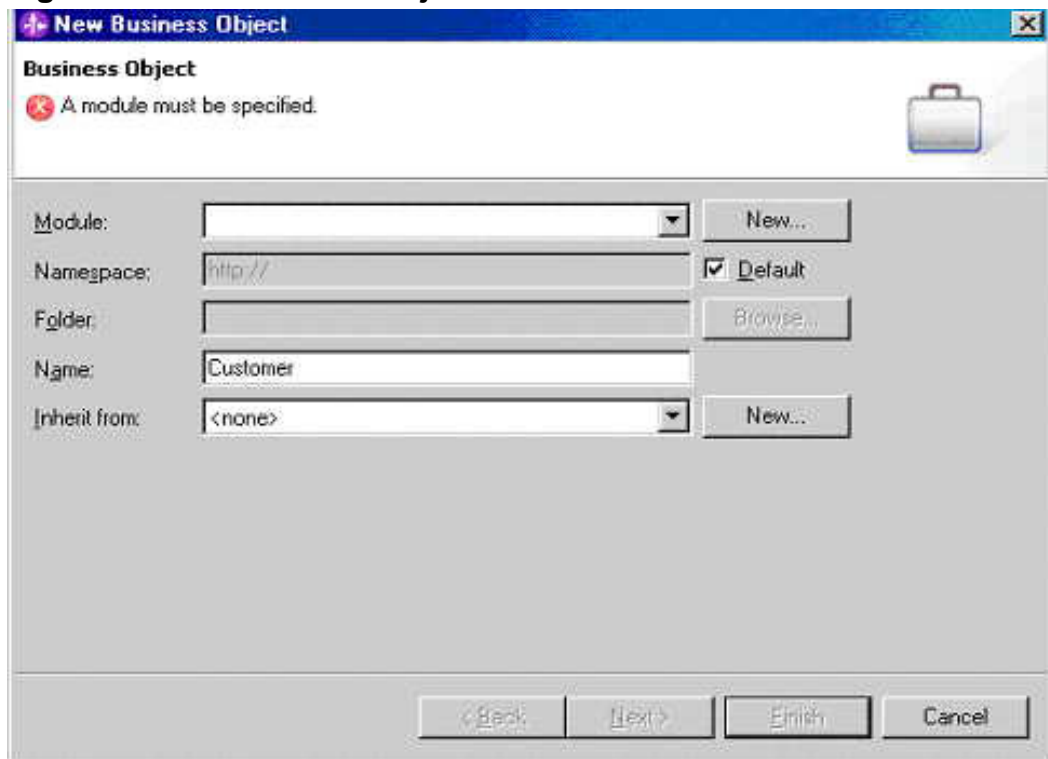
1. In the Windows task bar, select **Start > All Programs > IBM WebSphere > Integration Developer V6.0.1 > WebSphere Integration Developer V6.0.1**.
2. Wait until WebSphere Integration Developer starts.

Creating a business object for the message flow

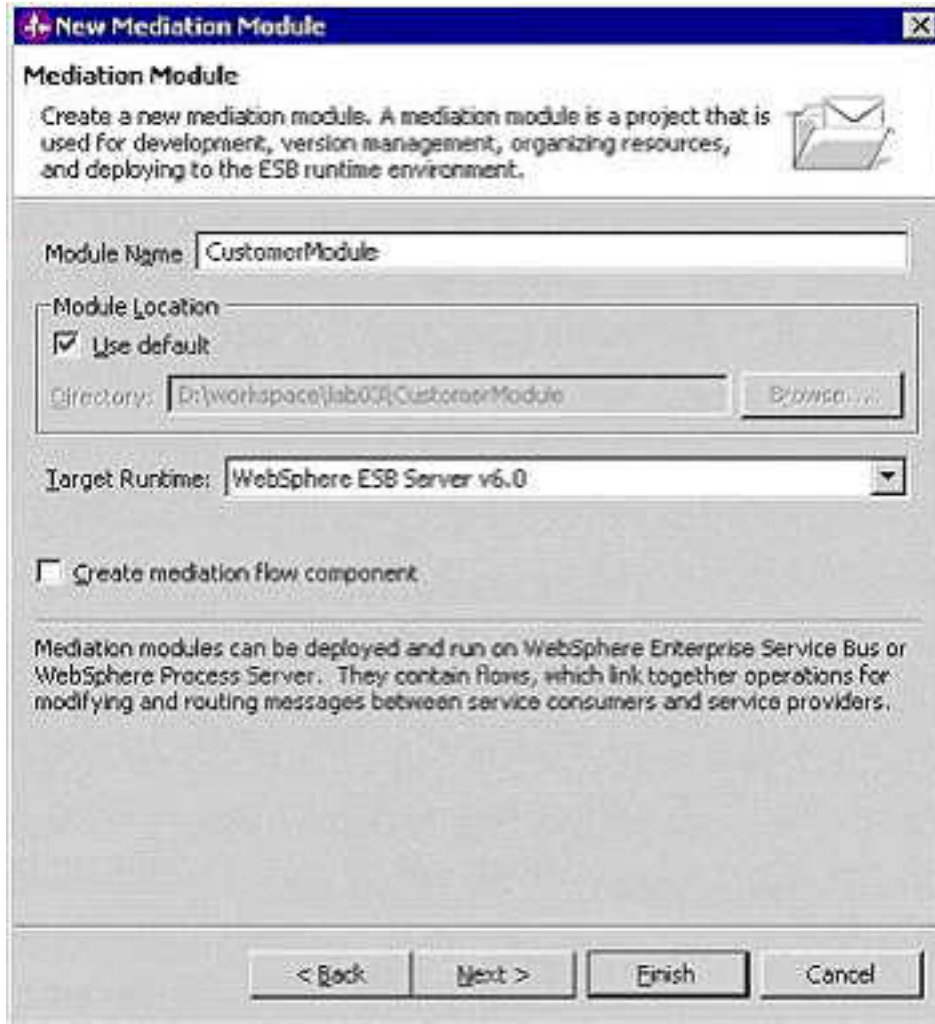
Use WebSphere Integration Developer to create a business object that is going to be used in the message flow:

1. In the Business Integration perspective, select **File > New > Business Object**.
2. Enter `Customer` in the Name field.

Figure 8. New Business Object window



3. Click **New** beside the Module field to create a mediation module that contains the business object.
 4. Select **Create a mediation module project** in the New Integration Project window. Click **Next**.
 5. Enter `CustomerModule` in the Module name field.
 6. Ensure **WebSphere ESB Server v6.0** is selected as the target runtime.
 7. Uncheck **Create mediation flow component**. Click **Finish**.
- Figure 9. New Mediation Module window**

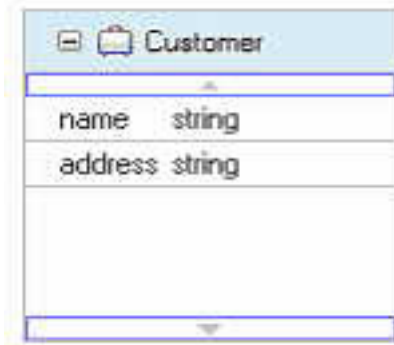


8. Click **Finish** again to close the New Business Object window and the Business Object editor should start automatically.
9. In the Business Object editor, click **Add Attribute** to add a new attribute to the business object.

Figure 10. Add attribute button



10. Enter name in the highlighted field.
11. Click **Add Attribute** again
12. Enter `address` in the highlighted field. The completed business object should look like the following:

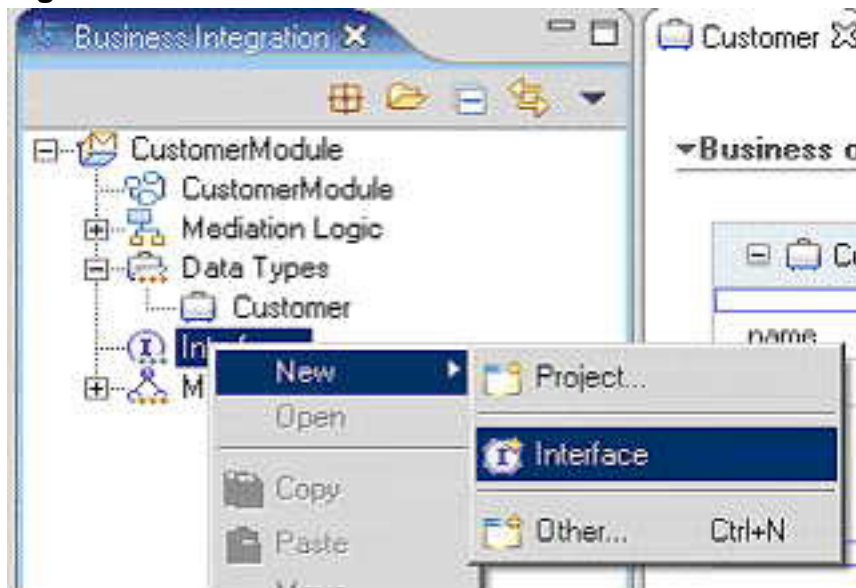
Figure 11. Customer business object

13. Press **Ctrl-S** to save the business object.

Creating the WSDL file for the flow input

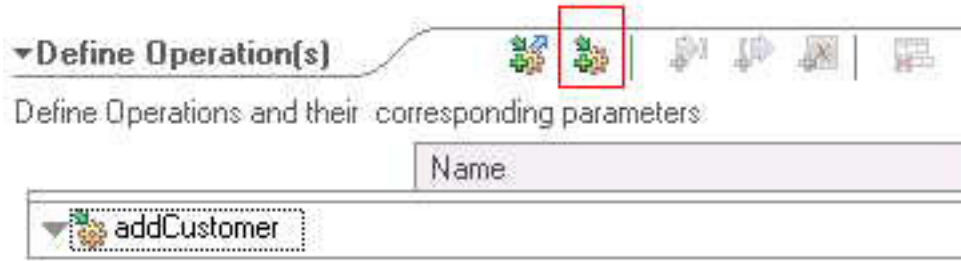
Use the WSDL editor in WebSphere Integration Developer to create an interface for the message flow.

1. In the Business Integration view, expand **CustomerModule**, right-click on **Interfaces** and select **New > Interface**. A New Interface Wizard window opens:

Figure 12. Create new interface

2. Enter `AddCustomer` in the Name field. Click **Finish**. This brings up the WSDL editor.

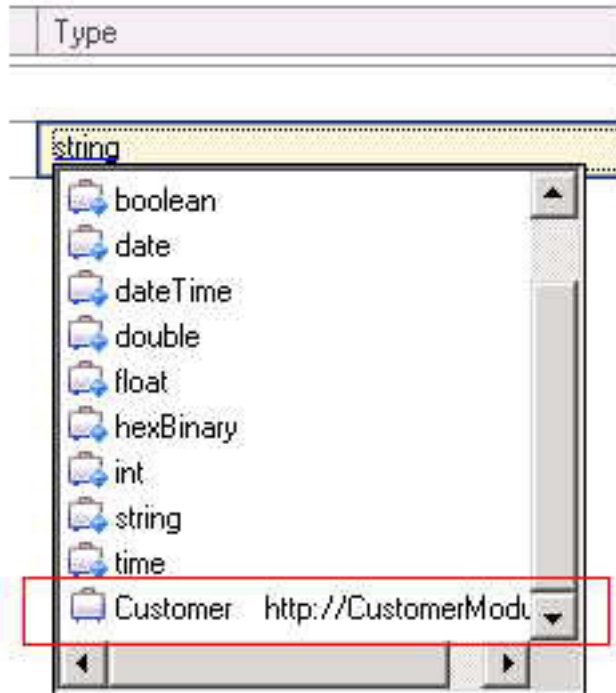
3. Click **Add One Way Operation** to add an operation.
Figure 13. Add One Way Operation button



4. Change the operation name to **addCustomer**.
5. Click the **Add Input** button to add an input message part. An input message is created automatically for the addCustomer operation.
Figure 14. Add Input button



6. Change the name to **customer**. This changes the message part name to customer.
7. Change the type to **Customer** by clicking on the input type, scrolling down and selecting **Customer** business object. This changes the message part type to Customer business object.
Figure 15. Select Customer business object



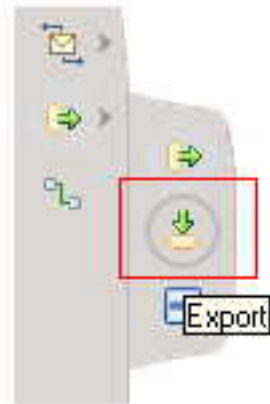
8. Press **Ctrl-S** to Save the WSDL file.

Using SCA to transform HTTP requests to JMS

Use SCA to transform HTTP requests to JMS.

1. In the Business Integration view, expand **CustomerModule** and double-click **CustomerModule** to open the Assembly Diagram editor.
2. Add an export component by clicking on the **export** icon in the palette of the Assembly Diagram editor, then click anywhere on the Assembly Diagram.

Figure 16. Export button



3. The default export component is named `Export1`. Right-click on the export component and select **Rename**. Enter `HTTPExport` as the new name.
4. Right-click on `HTTPExport`, then select **Add Interface**. An Add Interface window opens.
5. Select **AddCustomer** and click **OK**.
6. Right-click `HTTPExport` and select **Generate Binding > Web Services Binding**.
7. Click **Yes** on the Binding File Generation window.
8. Select **soap/http** and click **OK** on the Select Transport window.
9. Add an import component by clicking on the **import** icon in the palette of the Assembly Diagram editor, then click anywhere on the Assembly Diagram.

Figure 17. Import button

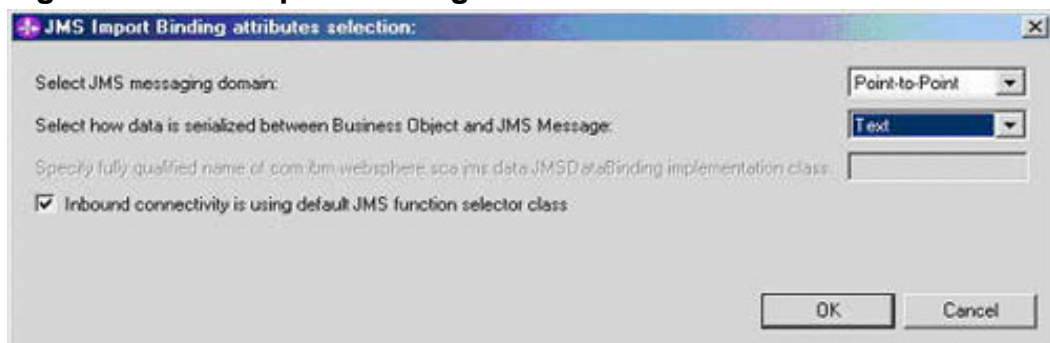


10. The default import component is named `Import1`. Right-click on the import component and select **Rename**. Enter `JMSImport` as the new name.

11. Wire the **HTTPExport** to the **JMSImport**. Click **OK** in the Add Wire window. Your diagram should look like the following:
Figure 18. Message flow



12. Right-click on **JMSImport** and select **Generate Binding > JMS Binding**.
13. In the window, select **Text** for the field Select how data is serialized between Business Object and JMS message.
Figure 19. JMS Import Binding attribute selection window



14. Click **OK**.
15. Press **Ctrl-S** to save the assembly diagram.

Configuring the bus

Configure the WebSphere Enterprise Service Bus test server for the message flow you created:

1. In the Servers view, start WebSphere Enterprise Service Bus server by right-clicking the **WebSphere ESB server v6.0** instance and select **start**. Wait until the server starts.
2. After the server starts, right-click on the server instance again and select **Run the administrative console** to start the administrative console.
3. In the welcome page of administrative console, enter `admin` for the User ID and click **Log in**. You can use any value for the User ID.
4. On the next page, click **Service integration > Buses** in the navigation

panel.

- In the Bus section, click on **SCA.APPLICATION.esbCell.Bus**.
Figure 20. Bus section

Select	Name	Description
<input type="checkbox"/>	CommonEventInfrastructure Bus	CommonEventInfrastructure Bus
<input type="checkbox"/>	SCA.APPLICATION.esbCell.Bus	Messaging bus for Service
<input type="checkbox"/>	SCA.SYSTEM.esbCell.Bus	Messaging bus for Service

- In the Destination resources section, click **Destinations**:
Figure 21. Destination resources section



- Click **New** to create a new destination.
- On the Create new destination page, select **Queue**, then click **Next**.
- Enter `JMSImportOut` in the identifier field. Click **Next**.
- On the Create new queue page, click **Next**.
- The Confirm queue creation page tells you that a new queue "JMSImportOut" will be created and a queue point for this queue will be created on the bus. Click **Finish** to confirm queue creation.
- Review the new queue JMSImportOut as it appears in the following destinations list:

Figure 22. Destination list

Select	Identifier	Type
<input type="checkbox"/>	Default.Topic.Space	Topic space
<input type="checkbox"/>	JMSImportOut	Queue
<input type="checkbox"/>	_SYSTEM.Exception.Destination.esbNode.server1-SCA.APPLICATION.esbCell.Bus	Queue

13. Click **save** and **save** again on the next page to apply the changes.
14. Click **Resources > JMS Providers > Default messaging** on the navigation panel.
15. A JMS queue connection factory is used to create connections to the associated JMS provider of JMS queues, for point-to-point messaging. In the Connection Factories section, click **JMS queue connection factory**.
16. Click **New** in the Default message provider section.
17. Enter `myBindingQCF` in the Name field
18. Enter `jms/myBindingQCF` in the JNDI name field
19. Select **SCA.APPLICATION.esbCell.Bus** in the Bus Name field. Your page should now look like the following:
Figure 23. Queue connection factory attributes

Administration

* Scope

* Name

* JNDI name

Description

Category

Connection

* Bus name

20. Click **OK** at the bottom of the page.
21. A queue connection factory is now created and you should see the following entry in the Default messaging provider page:

Figure 24. Default message provider page

Select	Name ↕	JNDI name ↕
<input type="checkbox"/>	myBindingQCF	jms/myBindingQCF

22. Click **save** here and again on the next page to apply the changes.

23. Click **Resources > JMS Providers > Default messaging** on the navigation panel.
24. A JMS queue is used as a destination for point-to-point messaging. In the **Destinations** section, click **JMS queue**.
25. In the Default messaging provider page, click **New**.
26. Enter `JMSImportOut` in the Name field.
27. Enter `jms/JMSImportOut` in the JNDI name field.
28. Select **SCA.APPLICATION.esbCell.Bus** in the Bus Name field. Make this selection first to enable the drop-down list in the Queue Name field.
29. Select **JMSImportOut** in the Queue Name field. Your page should now look like the following:

Figure 25. JMS Queue attributes

General Properties

Administration

* Scope

* Name

* JNDI name

Description

Connection

Queue name

Bus name

30. Click **OK** at the bottom of the page.
31. A JMS queue is now created and you should see the following entry in the Default messaging provider page:

Figure 26. JMS Queue displayed

<input type="button" value="New"/> <input type="button" value="Delete"/>		
Select	Name ↕	JNDI name ↕
<input type="checkbox"/>	JMSImportOut	jms/JMSImportOut

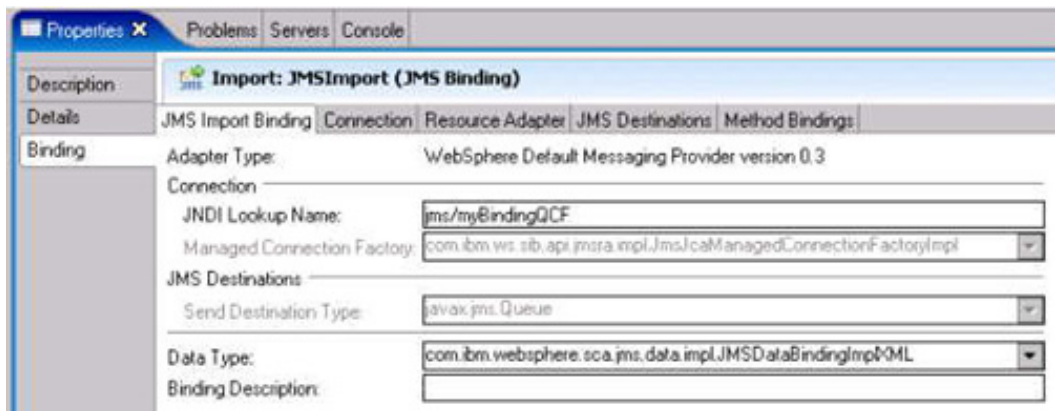
32. Click **save** here and again on the next page to apply the changes.
33. Click **Logout** to log out of the administrative console.

Configuring the mediation module to work on the bus

Configure the mediation module to work on the bus.

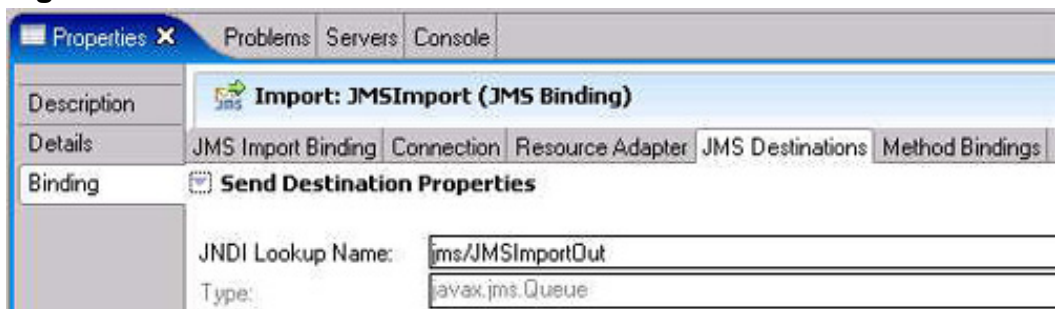
1. In the Assembly diagram editor, select **JMSImport**.
2. In the Properties view, select **Binding**.
3. Enter `.jms/myBindingQCF` in the JNDI Lookup Name field.

Figure 27. Binding tab

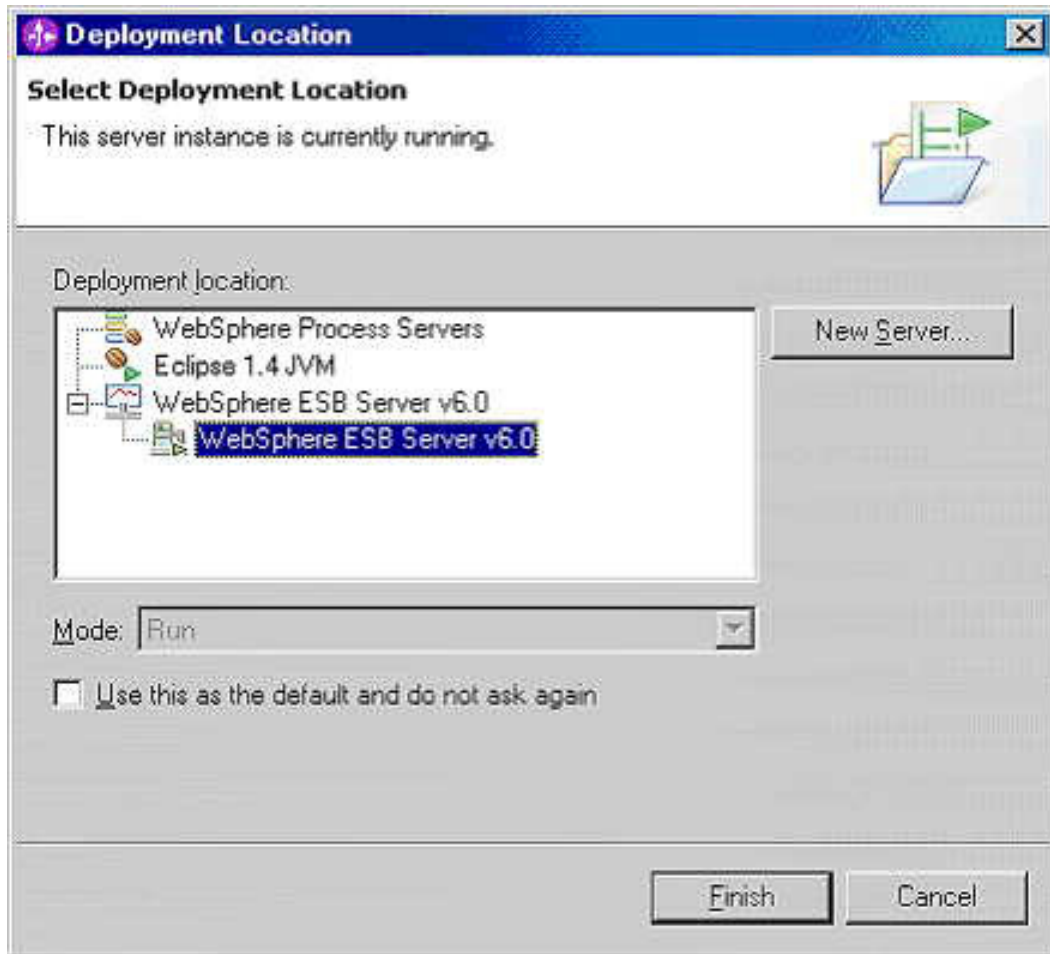


4. Select the **JMS Destinations** tab and expand the **Send Destination Properties**.
5. Enter `./jms/JMSImportOut` in the JNDI Lookup Name field.

Figure 28. JMS Destination tab



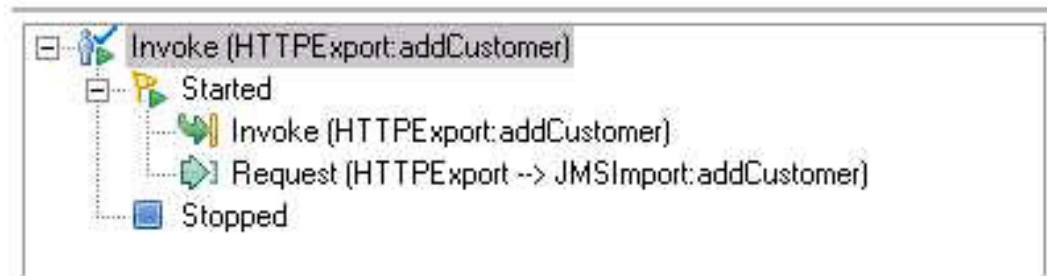
6. Press **Ctrl-S** to save the module.



8. Click **Finish** and wait until execution completes. The following should appear once the execution is completed:

Figure 31. Execution status

Events

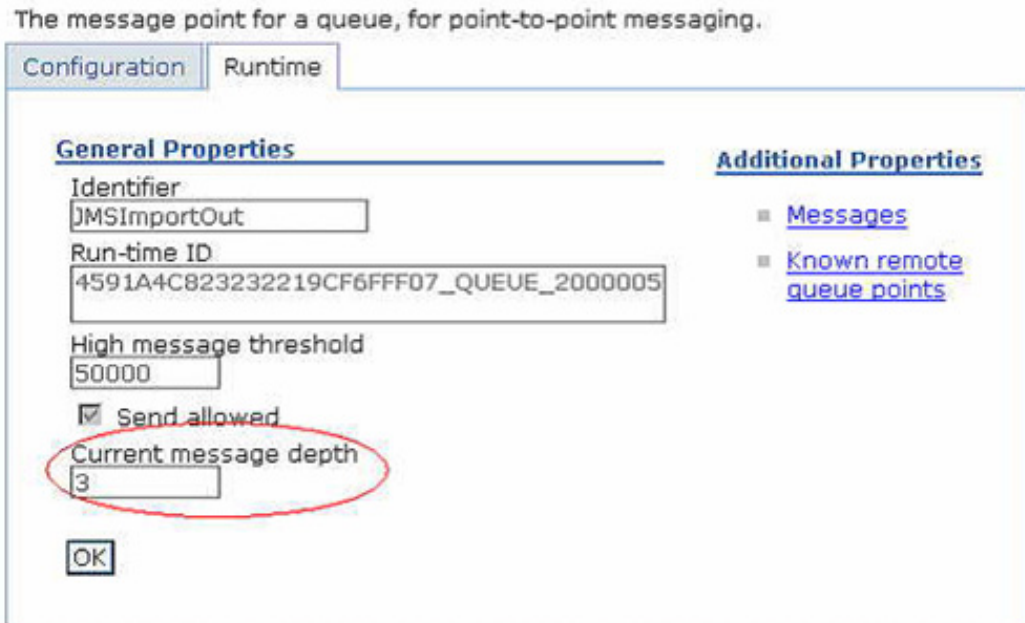


9. Look at the message in the queue. In the Servers view, right-click on the **WebSphere ESB server v6.0** instance and select **Run administrative console**.
10. On the welcome page of the administrative console, enter `admin` for the

User ID and click **Log in**. You can use any value for the User ID.

11. Select **Service integration > Buses > SCA.APPLICATION.esbCell.Bus > Destinations > JMSImportOut > Queue points > JMSImportOut@esbNode.server1-SCA.APPLICATION.esbCell.Bus**.
12. Click **Runtime** and the following page appears. The current message depth indicates how many messages are in the queue. This number increments by one each time a request is sent:

Figure 32. Current message depth



13. To see the messages in the queue, click **Messages**. A list of messages appears. The message with the highest identifier is the most recent message.

Figure 33. Message identifier

The screenshot shows a table with columns: Select, Position, Identifier, and State. There are three messages listed. The 'Identifier' for the third message, '3000008', is circled in red. Above the table are 'Delete' and 'Delete all' buttons. Below the table are icons for selection, copy, and other actions.

Select	Position	Identifier	State
<input type="checkbox"/>	1	3000000	Unlocked
<input type="checkbox"/>	2	3000007	Unlocked
<input type="checkbox"/>	3	3000008	Unlocked

- Click on the identifier link, and then click **Message body** to view the messages in the queue. Data shown in the red box is the data just sent.

Figure 34. Message in queue

* Displayed message body size
 1024 Bytes

0000000020:	3c3f786d	6c207665	7273696f	6e3d2231	2e302220	<?xml version="1.0"
0000000040:	656e636f	64696e67	3d225554	462d3822	3f3e0d0a	encoding="UTF-8"?.
0000000060:	3c637573	746f6d65	723a4375	73746f6d	65722078	<customer:Customer x
0000000080:	6d6c6e73	3a637573	746f6d65	723d2268	7474703a	mlns:customer="http:
0000000100:	2f2f4375	73746f6d	65724d6f	64756c65	223e0d0a	//CustomerModule"?.
0000000120:	20203c6e	616d653e	4a6f686e	20536d69	74683c2f	<name>John Smith</
0000000140:	6e616d65	3e0d0a20	203c6164	64726573	733e3832	name>.. <address>82
0000000160:	30302057	61726465	6e3c2f61	64647265	73733e0d	00 Warden</address>
0000000180:	0a3c2f63	7573746f	6d65723a	43757374	6f6d6572	</customer:Customer
0000000183:	3e0d0a					>..'

- Click **logout**.

Generating the Web Services proxy

Would you like to see these steps demonstrated for you?

 [Show me](#)

To generate a Web Services client to access the message flow created:

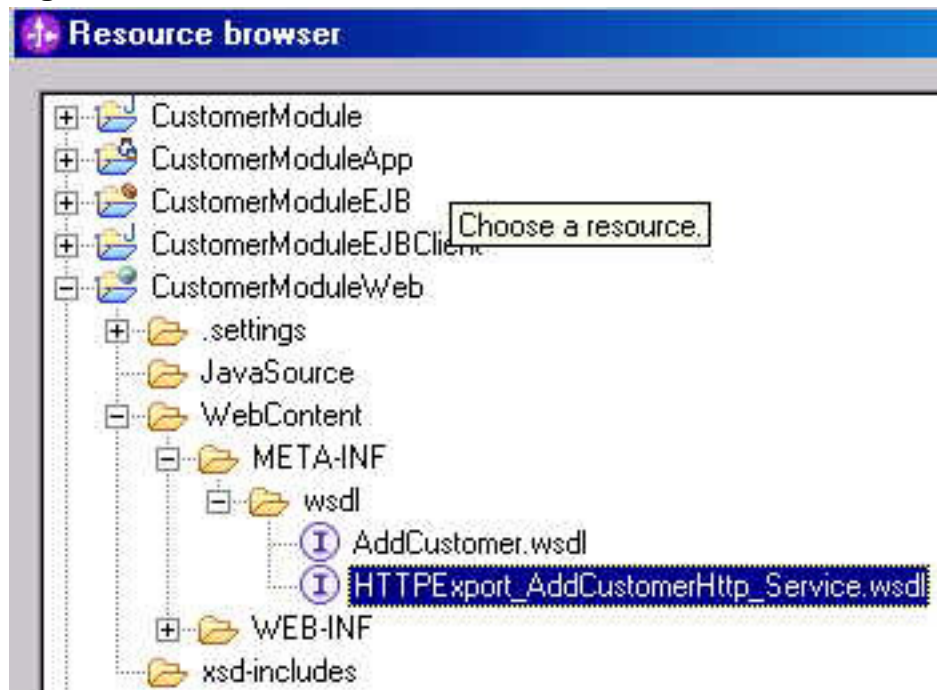
- In the Servers view, start WebSphere ESB server v6.0 if it's not already started.
- Make sure CustomerModuleApp is already added to the server using Add and remove projects.
- Select **File > New > Other....**
- In the New window, check **Show all Wizard**.
- Expand the **Web Service** folder, select **Web Service Client**, and then click **Next**. Click **OK** if a confirmation box appears.

Figure 35. New window



6. In the next window, make sure Overwrite files without warning is checked. Click **Next**.
7. Click **Browse...** and locate the HTTPExport_AddCustomerHttp_Service.wsdl file in the CustomerModuleWeb project. Click **OK**.

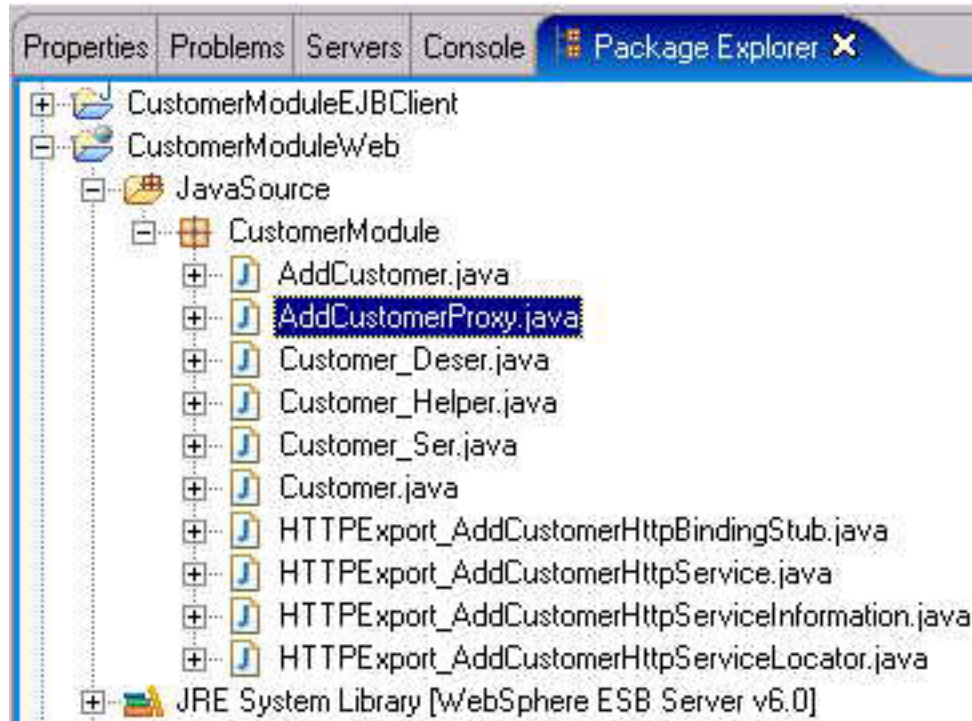
Figure 36. Resource browser



8. Click **Next**. In the next window, click **Next** again.
9. In the Web Service Proxy Page, click **Finish**.

10. The Web Service proxy is now generated. Go to the Java perspective.
11. In the Package Explorer view, expand **CustomerModuleWeb > JavaSource > CustomerModule**. The AddCustomerProxy and the corresponding files are the generated proxy.

Figure 37. Generated proxy

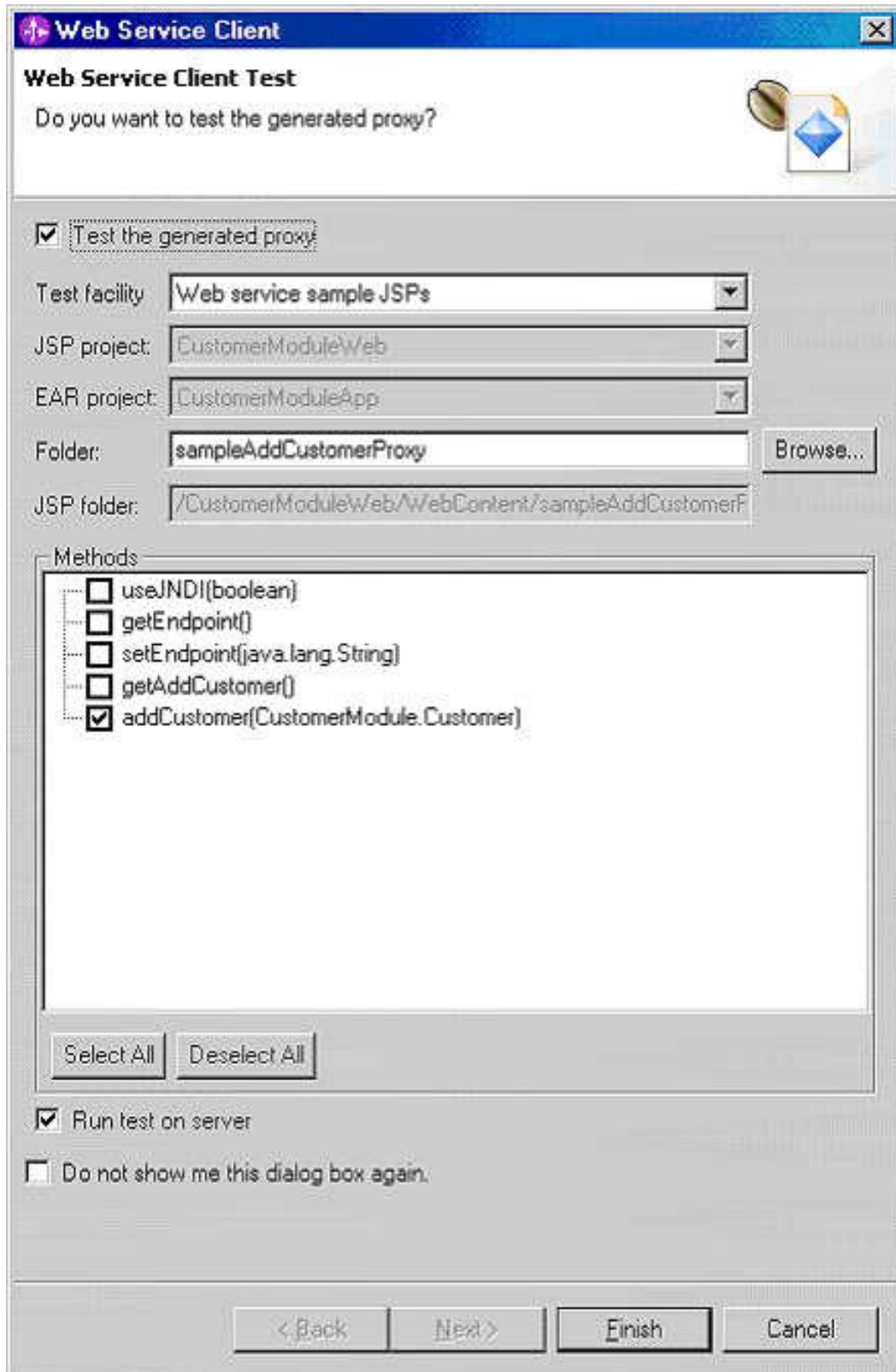


Generating a sample JSP and accessing it using a browser

To generate a sample JSP that accesses the Web Service proxy created in the previous step:

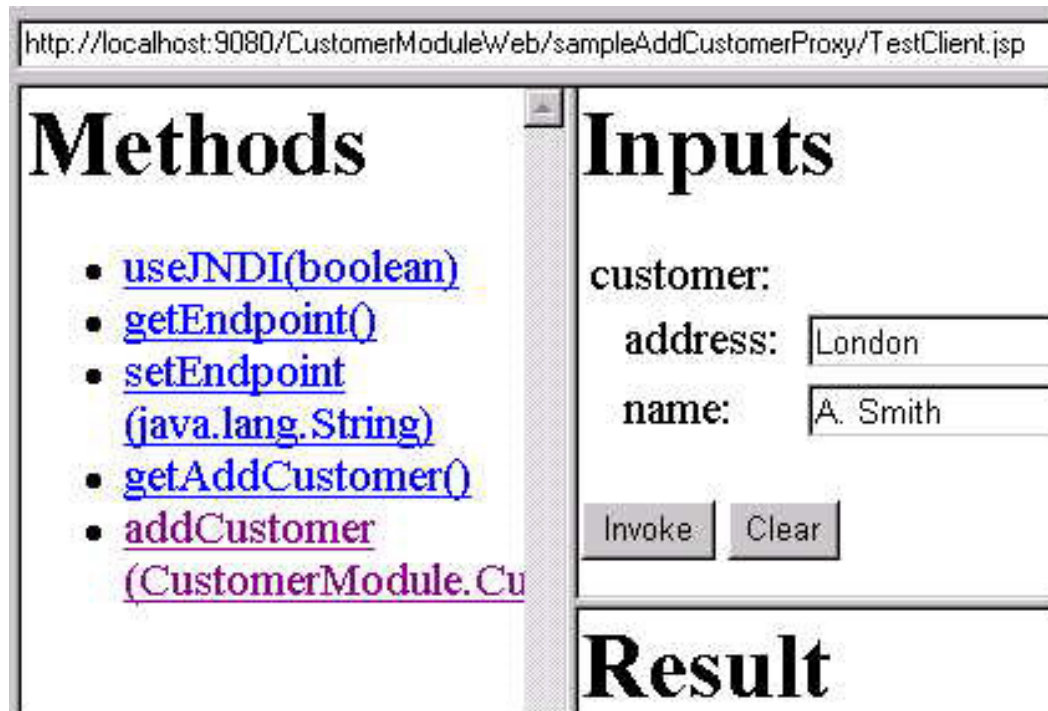
1. Select **AddCustomerProxy.java**.
2. Right-click, then select **WebServices > Generate Sample JSPs**.
3. In the Web Service Client page, uncheck **all methods except addCustomer** as shown below. Click **Finish**. The sample JSP is launched.

Figure 38. Web Service Client window



4. Select **addCustomer** in the Methods frame. Enter `London` in the address field and `A. Smith` in the name field. Click **Invoke**.

Figure 39. Sample page



5. Check the message in the queue by repeating steps 9 through 15 in [Testing using the integration test client](#).

Exporting the mediation module for deployment in WebSphere Enterprise Service Bus

Now you have a message flow that runs in the test server and is ready to deploy. Complete the following steps to export the message flow as an EAR file that can be deployed in WebSphere Enterprise Service Bus as a normal J2EE application:

1. In the Business Integration view, right-click on **CustomerModule** and select **Export....**
2. Select **EAR file** in the Export window. Click **Next**.
3. Select **CustomerModuleApp** in the EAR project filed.
4. Enter a location in the Destination field. Click **Finish**.

5. The EAR file is exported and ready for deployment.
-

Section 8. Summary

Congratulations! You have now completed Part 5 of the Hello, World! series. You've learned the basic concepts of ESB and how to build message flow for protocol transformation using SCA, use Integration Test Client in WebSphere Integration Developer, and generate a Web service proxy and sample JSP pages.

To keep up with all the Hello World tutorials and articles, check the [Hello World overview page](#).

Resources

Learn

- [Enterprise service bus](#), a book by David A. Chappell, describes the concept of ESB.
- [Redbook™: Getting Started with WebSphere Enterprise Service Bus V6](#).
- Learn about [WebSphere Enterprise Service Bus](#).
- Learn about [WebSphere Integration Developer](#).
- [Service Oriented Architecture \(SOA\)](#).
- [IBM SOA Foundation](#).
- [IBM WebSphere Business Process Management Version 6.0 Information Center](#).

Get products and technologies

- Build your next development project with [IBM trial software](#), available for download directly from developerWorks.

Discuss

- [Participate in the discussion forum for this content](#).
- Participate in [Global WebSphere Community blogs](#) and get involved in the community.

About the authors

Abelard Chow



Abelard Chow is an Advisory Software Engineer with the IBM Scenario Analysis Lab, where he works on improving the cross-brand integration capability of IBM Software Group products. In the last few years, he served IBM customers as a Solution Architect in IBM Financial Industry Solution.

Bill Zhu



Bill Zhu is a software developer in the Scenario Analysis Lab team at the IBM Toronto Lab. His areas of expertise include solution design and implementation, e-commerce software, and messaging systems.