

IBM WebSphere Commerce



Web Services Implementation Guide

Version 5.5

IBM WebSphere Commerce



Web Services Implementation Guide

Version 5.5

Note

Before using this information and the product that it supports, read the information in “Notices” on page 57.

First Edition, Second Revision (July 2004)

This edition applies to IBM WebSphere Commerce, Version 5.5 and to all subsequent releases and modifications of the above listed products, until otherwise indicated in new editions. Make sure that you are using the correct edition for the level of the product.

This edition applies to IBM WebSphere Commerce Studio, Business Developer Edition, Version 5.5 and IBM WebSphere Commerce Studio, Professional Developer Edition, Version 5.5 and to all subsequent releases and modifications of the above listed products, until otherwise indicated in new editions. Make sure that you are using the correct edition for the level of the product.

Order publications through your IBM representative or the IBM branch office that serves your locality.

IBM welcomes your comments. You can send your comments by using the online IBM WebSphere Commerce documentation feedback form, available at the following URL:

<http://www.ibm.com/software/commerce/rcf.html>

When you send information to IBM, you grant IBM a nonexclusive right to use or distribute the information in any way it believes appropriate without incurring any obligation to you.

© Copyright International Business Machines Corporation 2002, 2004. All rights reserved.

US Government Users Restricted Rights – Use, duplication or disclosure restricted by GSA ADP Schedule Contract with IBM Corp.

Contents

Before you begin	v
Knowledge requirements	vi
Conventions used in this guide	vi
Default paths	vi
Summary of changes	vii
Chapter 1. Introduction	1
Terminology	1
Overview	2
Why use Web services?	2
Web services architecture	3
Prerequisites	4
References	4
Chapter 2. Web services for WebSphere Commerce	7
Messaging infrastructure in WebSphere Commerce	7
Understanding the Web services implementation for WebSphere Commerce	7
WebSphere Commerce as a service provider	7
Processing the request	8
Response	8
Enabling WebSphere Commerce as a service provider	9
WebSphere Commerce as a service requestor	9
Chapter 3. Enabling WebSphere Commerce as a service provider	11
Infrastructure to enable Web services	11
Architecture	11
+ Enabling the HTTP program adapter	13
Security considerations	14
Web services and access control	14
Transport level security	14
Generating client code to access Web services	15
Sample Web services that can be invoked by external Web clients	15
Importing the sample code	16
OrderCreate service (Business Edition)	16
Enabling the OrderCreate sample Web service	16
Testing the sample	19
Security for the OrderCreate Web service	19
Creating users and contracts	20
Populating database tables	21
New message mappings	23
Configuring message mappers	24
OrderStatus service	25
Enabling the OrderStatus sample Web service	25
Testing the sample	28
Access control for the OrderStatus service	29
User roles for the OrderStatus Web service	29
The OrderStatus WSDL definition	29
New message mappings	30
Configuring message mappers	31
Disabling the OrderCreate and OrderStatus Web services	31
Chapter 4. Enabling WebSphere Commerce as a service requestor	35
Architecture	35
Finding a service implementation	36
Invoking a service implementation	37
Sample Web service that WebSphere Commerce can invoke on an external system	37
OrderFulfillment service	38
Interface WSDL definitions	38
Task commands	38
Business objects	40
Testing the sample	42
Disabling the OrderFulfillment Web service	42
Appendix A. Sample WSDL definition files	45
Appendix B. Using IBM WebSphere Studio Application Developer	47
Importing a WSDL into WebSphere Studio Application Developer	47
Generating the client code and data types	48
Testing the Web service	48
Using the TCP Monitor to trace SOAP messages	49
Creating a TCP Monitor server instance	50
Configuring the TCP Monitor server instance	50
Tracing the SOAP messages using the TCP/IP Monitor	50
Customizing the client code	54
Notices	57
Trademarks	58

Before you begin

The *Web Services Implementation Guide* is intended for those who want to expose the business processes in WebSphere® Commerce Version 5.5 Business Edition and Professional Edition as Web services. It also explains how WebSphere Commerce can access Web services that are hosted by external systems. This guide assists developers who need to understand how to enable and implement Web services for WebSphere Commerce. Other users such as system administrators who want to configure Web services and use the functionality provided by WebSphere Commerce can refer to this guide. Knowledge of the WebSphere Commerce programming model is a prerequisite. In addition, knowledge about invoking and implementing Web services and related protocols is required.

This guide provides information on how you can enable WebSphere Commerce to work with Web services and Web service clients. The guide provides an overview of Web services and its architecture. It briefly describes the messaging infrastructure in WebSphere Commerce, the WebSphere Commerce run time support to enable Web services, and related references. This guide also describes the sample Web services provided along with their associated business objects and task commands.

The guide is divided into the following sections:

Chapter 1. Introduction A brief overview of Web services as well as the definition of the terms used in this guide, prerequisites, and related references.

Chapter 2. Web services for WebSphere Commerce A brief overview of the messaging infrastructure in WebSphere Commerce and the Web services architecture for WebSphere Commerce. It describes the methodology to expose WebSphere Commerce business processes as Web services and the methodology that enables WebSphere Commerce to access Web services hosted by external systems.

Chapter 3. Enabling WebSphere Commerce as a Web services provider Describes how to enable a business process defined by WebSphere Commerce as a Web service using a sample definition. The new program elements introduced into WebSphere Commerce and the implementation details of the OrderCreate and the OrderStatus sample Web services are explained.

Chapter 4. Enabling WebSphere Commerce as a Web service requestor Describes how you can enable WebSphere Commerce to invoke a service defined by an external system using a sample definition. The new program elements introduced into WebSphere Commerce and the implementation details of the OrderFulfillment sample Web service are explained.

Appendix A. Sample WSDL definition files Lists the WSDL definition files for the sample Web services.

Appendix B. Using IBM® WebSphere Application Developer to create Web service client code Describes how you can use the IBM WebSphere Studio to generate the Web service client code.


Knowledge requirements

In order to use the information contained in this guide, you should have knowledge in the following areas:

- Web services technologies, such as SOAP, WSDL, and UDDI
- XML
- Java™
- JavaServer Pages technology
- WebSphere Studio Application Developer (Also referred to as WebSphere Studio in this document)
- WebSphere Commerce programming model

Conventions used in this guide



This guide uses the following conventions:

Boldface type	indicates commands or graphical user interface (GUI) controls such as names of fields, buttons, or menu choices.
monospaced type	indicates examples of text that you enter exactly as shown as well as file names and directory paths.
<i>Italic type</i>	is used for emphasis and for variables for which you substitute your own values.
	indicates information that is specific to WebSphere Commerce for AIX®.
	indicates information that is specific to WebSphere Commerce for IBM @server iSeries™ (formerly called AS/400®).
	indicates information that is specific to WebSphere Commerce for xSeries® Linux, information that is specific to WebSphere Commerce for @server zSeries® and S/390® Linux, information that is specific to WebSphere Commerce for @server iSeries Linux, and information that is specific to WebSphere Commerce for @server pSeries® Linux.
	indicates information that is specific to WebSphere Commerce for Solaris Operating Environment software.
	indicates information that is specific to WebSphere Commerce for Windows®.
	indicates information that is specific to DB2 Universal Database™.
	indicates information that is specific to the Oracle® database.
	indicates information that is specific to WebSphere Commerce Business Edition.
	refers to WebSphere Commerce Studio, Version 5.5 (Professional or Business Developer Editions).

Default paths

This guide uses the following default installation paths:

WC_installdir This indicates the installation path for WebSphere Commerce. When you see this variable, substitute the installation path for your installation of WebSphere Commerce. For example, substitute

	C:\IBM\WebSphere\CommerceServer55
	/usr/WebSphere/CommerceServer55

▶ Linux	/opt/WebSphere/CommerceServer55
▶ Solaris	/opt/WebSphere/CommerceServer55
▶ 400	/QIBM/ProdData/CommerceServer55

WAS_installdir This indicates the installation path for WebSphere Application Server. When you see this variable, substitute the installation path for your installation of WebSphere Application Server. For example, substitute

▶ Windows	C:\IBM\WebSphere\AppServer
▶ AIX	/usr/WebSphere/AppServer
▶ Linux	/opt/WebSphere/AppServer
▶ Solaris	/opt/WebSphere/AppServer
▶ 400	/QIBM/ProdData/WebAS5/Base

WCStudio_installdir This indicates the installation path for WebSphere Commerce Studio. When you see this variable, substitute the installation path for your installation of WebSphere Commerce Studio. For example, substitute

▶ Studio	C:\IBM\WebSphere\CommerceStudio55
----------	-----------------------------------

WCStudio_workspacedir is the directory for the WebSphere Commerce Studio workspace. For example, substitute

▶ DB2	C:\WebSphere\workspace_db2
▶ Oracle	C:\WebSphere\workspace_oracle

WAS_userdir This indicates the directory for all the data that is used by WebSphere Application Server which can be modified or needs to be configured by the user. When you see this variable, substitute,

▶ 400	/QIBM/UserData/WebAS5/Base/ <i>WAS_instance_name</i>
-------	--

where, *WAS_instance_name* is the variable that represents the name of the WebSphere Application Server with which your WebSphere Commerce instance is associated.

Summary of changes

- + The most recent version of this document is available as a PDF file from the
- + Technical Library page of the WebSphere Commerce Web site:
- + <http://www.ibm.com/software/commerce/library/>

+ To learn about last-minute changes to the product, see the current product
+ README file, also available from the WebSphere Commerce Web site.

+ This book uses the following convention for revision characters:

- + • The "+" (plus) character indicates updates that have been made in the current
+ revision of this document.
- + • The "|" character identifies cumulative updates that have been made in all
+ previous revisions of this document.

+ The following table shows the main changes that have been made to this book:

Change	Chapters or pages effected
Enabling the HTTP program adapter	Chapter 3, "Enabling WebSphere Commerce as a service provider," on page 11
Enabling the OrderCreate sample Web service	"OrderCreate service (Business Edition)" on page 16
Testing the sample	"OrderCreate service (Business Edition)" on page 16
Creating users and contracts	"OrderCreate service (Business Edition)" on page 16
Enabling the OrderStatus sample Web service	"Enabling the OrderStatus sample Web service" on page 25
Testing the sample	"OrderStatus service" on page 25
Generating the client code and data types	"Generating the client code and data types" on page 48
Testing the Web service	"Appendix B. Using IBM WebSphere Studio Application Developer" on page 47

Chapter 1. Introduction

This chapter gives an overview of Web services. It defines the terms used in this guide, lists prerequisites, and provides references to other related sources of information.

Terminology

The following terms are used in this guide:

Service provider

A service provider is the owner of the Web service. The service provider creates a Web service, publishes the service either to a service registry or directly to the service clients, and hosts the platform that provides access to the service.

Service requestor

A service requestor is the application that initiates an interaction with a Web service. One example of a service requestor would be a typical Internet browser. Another example is an application that uses Web services to gather information for its processing requirements. The service requestor binds to the service using the published information and invokes the service.

SOAP (Simple Object Access Protocol)

SOAP defines a model for using simple request and response messages that are written in XML as the basic protocol for electronic communication. It is a transport mechanism that is independent of the underlying platform and protocol. It facilitates publish, find, bind, and invoke operations.

UDDI (Universal Description, Discovery, and Integration)

The UDDI repository stores the descriptions of Web services. UDDI is a specification that defines a service registry of available Web services, serving as a collection of global electronic yellow pages. UDDI enables a company to publish a description of available goods and services to the registry, announcing itself as a service provider.

Web service

A Web service is a software component based on a published interface that describes a collection of operations that are network accessible using standardized XML messaging.

WSDL (Web services Description Language)

WSDL provides a way of describing the specific interfaces of Web services and application programming interfaces (APIs).

XML mapping template

XML mapping templates (or mapping templates) are used to process inbound XML messages and translate them to WebSphere Commerce command parameters. The mapping template file in WebSphere Commerce defines how to map the elements of an incoming XML message into the target command parameters.

For more information on the mapping template, refer to the WebSphere Commerce Production and Development online help.

XML message mapper

The XML message mapper in WebSphere Commerce is responsible for converting the incoming XML message into the Java object that the target command expects. A message mapper can be registered with the *instance_name.xml* configuration file and is uniquely identified by a message mapper ID. The XML message mapper uses the mappings defined in the mapping template file to convert the XML message into the corresponding Java object command parameter.

Note: *instance_name* is the name of the WebSphere Commerce instance.

For more information about the message mapper, refer to the XML message mapper information in the WebSphere Commerce Production and Development online help.

Overview

Web services are a new breed of Web applications. They are self-contained, self-describing, modular applications that can be published, located, and invoked across the Web. Web services perform functions that can be invoked ranging from a simple request to complicated business processes. Once a Web service is deployed and registered, other applications can discover and invoke the deployed service. The foundation for Web services are standards such as simple object access protocol (SOAP), the Web services description language (WSDL), and the Universal Description, Discovery, and Integration (UDDI) registry.

WebSphere Commerce Version 5.5 includes code and documentation that allows you to enable WebSphere Commerce business functions as Web services. You can allow WebSphere Commerce to be the service provider by enabling its business functions as Web services that can be accessed by external systems. You can also allow WebSphere Commerce to be the service requestor by enabling it to invoke Web services that are hosted by external systems.

Why use Web services?

Web services allow applications to be integrated more rapidly, easily and less expensively than ever before. Integration occurs at a higher level in the protocol stack, based on messages entered more on service semantics and less on network protocol semantics, thus enabling loose integration of business functions. These characteristics are ideal for connecting business functions across the Web, both between multiple enterprises and within a single enterprise. They provide a unifying programming model so that application integration both inside and outside the enterprise can be done with a common approach, leveraging a common infrastructure. The integration and application of Web services can be done in an incremental manner, using existing languages and platforms and by adopting existing applications.

A Web service is an interface that describes a collection of operations that are accessible through the network by using standardized XML messaging. A Web service is described using a standard, formal XML notation, called its service description. The service description includes all the details necessary to interact with the service, including message formats (that detail the operations), transport protocols and location. The interface hides the implementation details of the service, allowing it to be used independently of the hardware or software platform on which it is implemented and also independently of the programming language in which it is written.

WebSphere Commerce is designed to support Web services, both as a provider of Web services and as a requestor (client) to other Web services. For example, external Web service clients can connect to WebSphere Commerce and use the Web services provided (as displayed in Figure 1). Alternatively, WebSphere Commerce can connect to other Web service providers and request information (as displayed in Figure 2).

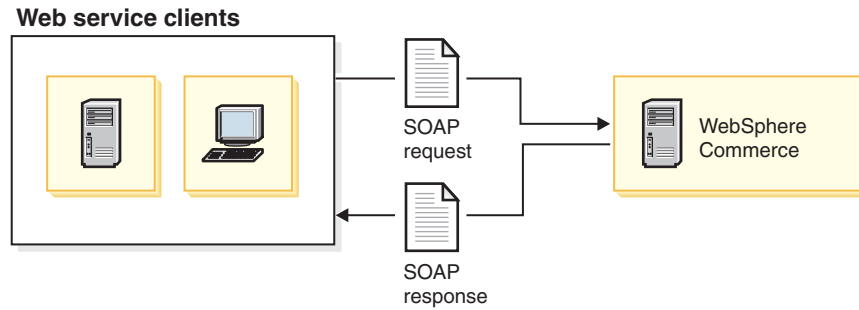


Figure 1. WebSphere Commerce as a service provider

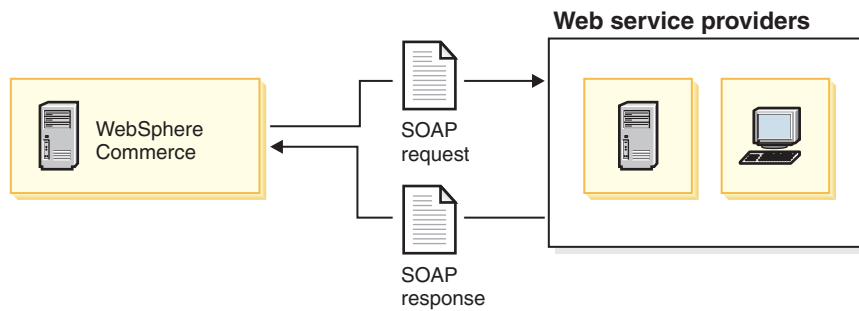


Figure 2. WebSphere Commerce as a service requestor

Web services architecture

The Web services architecture describes three roles: service provider, service requester, and service broker; and three basic operations: publish, find, and bind. A network component can play any or all of these roles. The service providers publish Web services to a service broker. Service requestors find required Web services by using a service broker and bind to them.

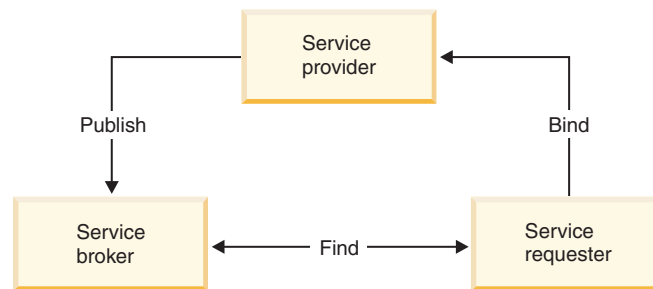


Figure 3. Web services architecture

In a typical scenario, a service provider hosts a network-accessible service module, which is an implementation of a Web service. The service provider defines a

service description for the Web service and publishes it to a service requestor or a service registry. The service requestor uses a find operation to retrieve the service description locally or from the service registry hosted by the service broker. The service requestor uses the service description to bind with the service provider and invoke or interact with the Web service implementation.

This solution provides run-time support to enable business processes that are defined by WebSphere Commerce as Web services. For samples on how you can use existing WebSphere Commerce commands as Web services, see Chapter 3, “Enabling WebSphere Commerce as a service provider,” on page 11. For samples on how WebSphere Commerce can invoke Web services that are defined by external systems, see Chapter 4, “Enabling WebSphere Commerce as a service requestor,” on page 35.

Prerequisites

Before using the information provided in this guide, ensure that you have installed IBM WebSphere Commerce Version 5.5. When installing WebSphere Commerce Version 5.5 you must select the **Custom** installation option to install the sample Web services provided. You will be asked to select the components that you want to install. Select **WebSphere Commerce example files**. For more information refer to the *WebSphere Commerce Installation Guide*.

Alternatively, if you are using WebSphere Commerce Studio to enable the sample Web services, then ensure that you have completed installing WebSphere Commerce Studio Version 5.5. For more information refer to the *WebSphere Commerce Studio Installation Guide*.

If you wish to modify the sample Web services, then ensure that you are using WebSphere Commerce Studio as your development environment.

References

For more information on Web services and related technologies, you can refer to the following Web sites:

- For information on Apache SOAP, an implementation of the SOAP submission to W3C, refer to <http://ws.apache.org/soap/>
- For in-depth information about the SOAP Specification, refer to <http://www.w3.org/TR/SOAP/>
- For information on UDDI, refer to <http://www.uddi.org/>
- For information on WSDL, refer to <http://www.uddi.org/specification.html>
- To learn more about Web services from IBM Alphaworks, refer to <http://www.alphaworks.ibm.com/webservices>
- To learn more about Web services from the perspective of developers, refer to <http://www.ibm.com/developerworks/webservices>
- For complete information about IBM WebSphere Commerce Business Edition software, refer to http://www.ibm.com/software/webservers/commerce/wc_be/
- For complete information about IBM WebSphere Commerce Professional Edition software, refer to http://www.ibm.com/software/webservers/commerce/wc_pe/
- For information on support for business partners to develop superior WebSphere Web services solutions, refer to <http://www.ibm.com/websphere/wow>

- For information on IBM WebSphere Studio, refer to www.ibm.com/software/webservers/commerce/commercestudio/

Note: The preceding Web addresses can change at any time without notice. IBM is not responsible for the authenticity or correctness of information from non-IBM Web sites.

Chapter 2. Web services for WebSphere Commerce

This chapter briefly explains the messaging infrastructure in WebSphere Commerce and helps you understand the Web services architecture for WebSphere Commerce. It also describes the methodology to expose WebSphere Commerce business processes as Web services and how WebSphere Commerce can access Web services hosted by external systems.

Messaging infrastructure in WebSphere Commerce

The WebSphere Commerce run-time architecture includes protocol listeners, adapters, controllers, commands, and views. The messaging infrastructure in WebSphere Commerce uses these components.

A protocol listener is a WebSphere Commerce run-time component that receives inbound requests from transports and then dispatches the requests to the appropriate adapters, based on the protocol used.

In WebSphere Commerce Version 5.5, the `RequestServlet` serves as the protocol listener that receives the SOAP version 1.1 messages. When the `RequestServlet` receives an XML request, it passes the request to the adapter manager. The adapter manager then queries the adapter types to determine which adapter can process the request. Once the specific adapter is determined, the request is passed to that adapter.

The program adapter is invoked when XML messages are received over the HTTP protocol. It allows external systems to communicate with WebSphere Commerce by passing XML requests over the HTTP protocol. The program adapter provides external systems such as procurement systems, supplier systems, and others with a common way to communicate with WebSphere Commerce over the HTTP protocol. In this way, WebSphere Commerce seamlessly integrates with external systems.

For more information about the flow of a request, refer to the *WebSphere Commerce Programming Guide and Tutorials*.

Understanding the Web services implementation for WebSphere Commerce

The WebSphere Commerce programming model consists of URL commands, controller commands, task commands, view commands, a program adapter, a Web controller, and more. For details about the WebSphere Commerce programming model, refer to the *WebSphere Commerce Programming Guide and Tutorials*.

The following sections explain how WebSphere Commerce fits into the service provider and service requestor roles:

WebSphere Commerce as a service provider

When you enable business processes in WebSphere Commerce as Web services that can be accessed by external systems, WebSphere Commerce becomes the service provider. WebSphere Commerce Version 5.5 includes enhancements to the

messaging system to support SOAP messages and responses. Additionally, the samples provided show how you can expose existing WebSphere Commerce business logic as Web services.

A client can invoke a URL command over the network, using the HTTP or secure HTTP protocols. WebSphere Commerce can also receive XML messages over the HTTP and HTTPS protocols or over the messaging middleware. This capability is further extended to allow WebSphere Commerce to receive SOAP messages over HTTP and this functionality is used to enable Web services. The `RequestServlet` is the entry point for all URL requests into the WebSphere Commerce system. When the `RequestServlet` receives an XML message, it identifies the content type as XML and invokes the program adapter.

Processing the request

The program adapter provides support for remote programs that invoke WebSphere Commerce commands. It receives requests and uses a message mapper to convert the request into an appropriate input to a command. After the conversion, the program adapter executes the command.

In WebSphere Commerce Version 5.5, a new message mapper and template mapping file are added to the WebSphere Commerce run-time architecture, to facilitate the parsing and processing of SOAP XML messages.

This new template-mapping file contains the mappings between the incoming SOAP XML message and the command parameters. The new message mapper looks into this template mapping file and parses the SOAP XML message into an appropriate input to a command, and then passes the input back to the program adapter. The program adapter then invokes the Web controller, which in turn executes the command specified by the message mapper.

Response

In WebSphere Commerce, when the controller command is executed successfully, it returns the name of a view to be executed. The Web controller invokes the view command for the corresponding view and composes the message to be sent back. There are a number of ways to form a response view. These include redirecting to a different URL, forwarding to a JSP template or writing an HTML document to the response.

In order to generate messages for different kinds of client devices, WebSphere Commerce supports different device formats. The Web controller picks up a view name corresponding to the device format of the incoming message. In WebSphere Commerce Version 5.5, a new device format, `SOAPHTTP` is introduced for the SOAP XML message. A new set of JSP templates can be defined for the same view names, but with a new device format. These JSP templates compose the SOAP response to be returned to the SOAP client. This follows the model view controller pattern and leverages the separation of the view from the business logic.

A new view command implementation is provided to handle the SOAP responses. The interface for this view command implementation is named `com.ibm.commerce.me.soap.commands.SOAPViewCommand` and the implementation class name is `com.ibm.commerce.me.soap.commands.HttpSOAPViewCommandImpl`. Therefore, whenever a new JSP template is registered in the `VIEWREG` table to compose the SOAP response message, this interface name and class name should be used.

The new SOAP view command implementation can handle both HTTP redirect and HTTP forward requests. Typically, HTTP redirects are handled by a URL redirection. Since a SOAP client cannot understand a URL redirection, this SOAP view command implements the HTTP redirect also as an HTTP forward.

Enabling WebSphere Commerce as a service provider

The following is an overview of the steps that allow you to make a WebSphere Commerce business process available as a Web service:

1. Identify the business logic that you want to expose as a Web service.
2. Identify the URL command that implements this business logic. If it is not available, then create a new URL command. You can write new commands to call a collection of existing commands.
3. Identify the mandatory and optional parameters that this URL command requires.
4. Define the WSDL definition for the Web service.
5. Map the incoming SOAP request to the command parameters by using the mapping template file. Ensure that the TemplateTag contains mappings for all the elements present in the WSDL definition. This requires that you populate the data for all the elements in the SOAP request when sending the request from the client. The mapping template file defines how each element in the incoming SOAP XML message is mapped to a command parameter using XPATH mappings. You can intercept the SOAP request using a utility like the TCP Monitor and then use the intercepted message to define the map.

Note: The TCP Monitor is available with WebSphere Studio Application Developer 5.0, refer to the WebSphere Studio help for more information. A stand alone version of the TCP Monitor is shipped with the Apache AXIS distribution. See, <http://ws.apache.org/axis/> for details.

6. Write a JSP template to compose a response.
7. Deploy the service. This includes the deployment of related files and resources. In case of a new command, you must deploy and register it in the WebSphere Commerce command registry.

Business logic in WebSphere Commerce is implemented using controller commands. A controller command is a command that interacts directly with the Web controller. A URL command is a controller command that can be invoked using a URL. As a result you can invoke a URL command by entering a URL in a Web browser. Before implementing a Web service, identify the URL command that implements the business logic, which you want to expose as a Web service. For example, the OrderStatus command allows you to update the status of an existing order in WebSphere Commerce.

For more information about URL commands, refer to the Reference section in the WebSphere Commerce Production and Development online help.

WebSphere Commerce as a service requestor

When you use WebSphere Commerce to initiate a Web service that is hosted by an external system, WebSphere Commerce becomes the service requestor. WebSphere Commerce Version 5.5 includes samples that show how WebSphere Commerce business logic can be integrated with other systems by making Web service calls. WebSphere Commerce sends Web service requests through the Web service proxies, which send and receive SOAP-based requests and responses. The business logic will work based on the identified Web service binding interfaces. The samples provided support the invocation of a Web service based on these interfaces.

The implementations can either be published in a service registry or stored in the local file system. Invoking a Web service from WebSphere Commerce includes the following activities:

1. Identify the published Web service interface definition, which suits the existing business scenarios that WebSphere Commerce supports.
2. Download the WSDL definition of the service implementation into your local WebSphere Commerce machine.
3. Register the Web service definition file in the *instance_name.xml* configuration file (where *instance_name* is the name of your WebSphere Commerce Server instance).
4. Create client proxies that WebSphere Commerce can use. You can generate them using tools such as IBM WebSphere Studio. When you generate the client proxy for the Web service interface definition, the data types required for the input messages and the output messages are also generated. See, the “Generating client code to access Web services” on page 15 section of this document for more information.

You can also use other tools to generate the client proxy, for example, the Web Services Toolkit. Refer to, <http://www.alphaworks.ibm.com/webservices>

5. Write a task command that implements the logic to do the following:
 - a. Retrieves the access points.
 - b. Invokes the proxy for the Web service.
 - c. Processes the results returned from the Web service.
6. Update the WebSphere Commerce command registry to register the task command.

Chapter 3. Enabling WebSphere Commerce as a service provider

This chapter describes how to enable a business process defined by WebSphere Commerce as a Web service. In this scenario, WebSphere Commerce acts as the service provider to external systems. Samples are provided that demonstrate how to enable the OrderCreate and the OrderStatus business processes as Web services.

Infrastructure to enable Web services

The following are the WebSphere Commerce run-time elements to support an incoming Web service request:

- SOAP message mapper
- Web services mapping template
- SOAP response view commands

External users can invoke WebSphere Commerce services by using a client program. These users can connect from any system that is capable of sending and receiving SOAP messages. WebSphere Commerce version 5.5 supports SOAP messages over the HTTP transport protocol.

The WebSphere Commerce services should be published as WSDL definitions. Web service clients can download the WSDL definitions, generate a client, and call the WebSphere Commerce Web Service using SOAP over HTTP.

Architecture

The message flow in this architecture is synchronous. This means that the external system waits for the response from the WebSphere Commerce system before proceeding with further processing. The inbound messaging system is used to allow external clients to invoke WebSphere Commerce business processes that are published as Web services.

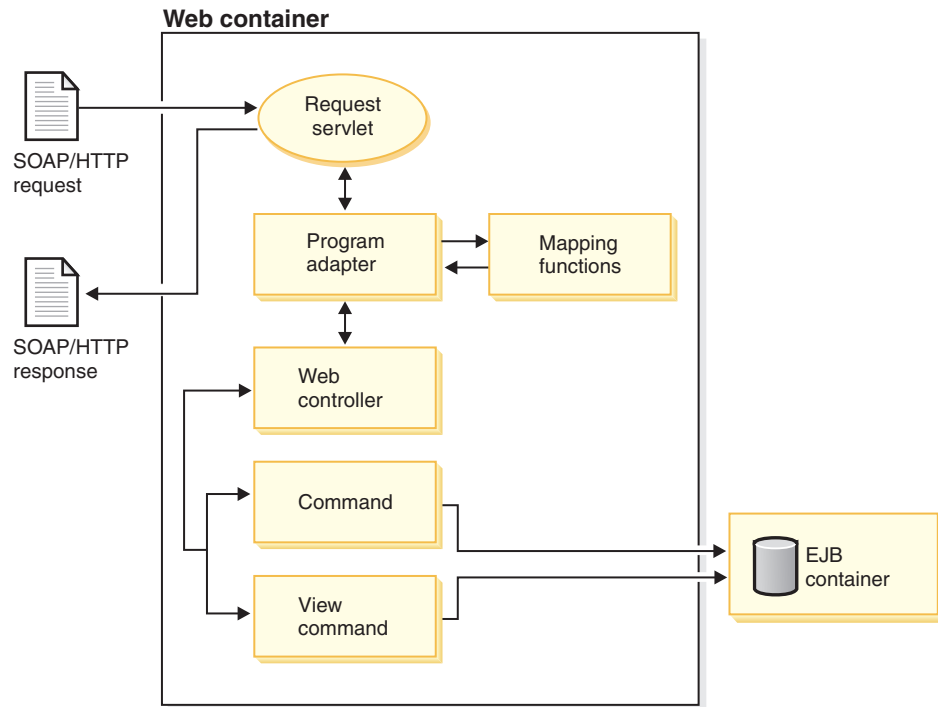


Figure 4. Architecture to enable WebSphere Commerce as a service provider

The following list describes the interaction when an external system invokes a WebSphere Commerce business process using a Web service:

1. The external system sends the well-formed SOAP request to WebSphere Commerce.
2. The RequestServlet receives the request and delegates it to the program adapter.
3. The program adapter invokes the registered SOAP message mapper.
4. The SOAP message mapper parses the SOAP messages and returns the name-value pairs to the program adapter using a template file, which contains the mapping information between the parameters in the Web service request and the command parameters.
5. The program adapter calls the Web controller and passes the command name and the name-value pairs to the Web controller.
6. The Web controller executes the command. In the examples provided, the BatchOrderRequest and the OrderStatusCmd commands are invoked.
7. After the command is executed the view name is returned to the Web controller.
8. The Web controller executes the registered view command to form the SOAP response and returns it to the program adapter.
9. The HTTP program adapter returns the message to the RequestServlet.
10. The RequestServlet sends the response to the waiting external client.

The following are the samples added to support the Web service provided by the service provider:







- The `webservice_template.xml` file is provided to map the XPath paths with the command properties.
- The following sample JSP templates are provided to compose the SOAP response and error messages:

- SOAP_OrderStatusResponse.jsp
- SOAP_PurchaseOrderResponse.jsp
- SOAP_GenericApplicationError.jsp
- SOAP_GenericSystemError.jsp

Enabling the HTTP program adapter

Invoking the WebSphere Commerce Web services requires the HTTP program adapter. Do the following to enable the HTTP program adapter:

1. Open the *instance_name.xml* file in a text editor from

-  *WC_installdir\instances\instance_name\xml*
-  *WCstudio_installdir\Commerce\instances\instance_name\xml*
-    *WC_installdir/instances/instance_name/xml*
-  */QIBM/UserData/CommerceServer55/instances/instance_name/xml*

2. Locate the following section:

```
<MessageMapper classname="com.ibm.commerce.messaging.programadapter.
  messagemapper.soapecsax.ECSAXMessageMapper"
  enable="false"
  messageMapperId="-3"
  name="WCS.SOAPINTEGRATION">
  <configuration EcInboundMessageDtdFiles=""
    EcInboundMessageDtdPath="WC_installdir/xml/messaging"
    EcSaxParserClass="com.ibm.xml.parsers.SAXParser"
    EcSystemTemplateFile="webservice_template.xml"
    EcTemplatePath="WC_installdir/xml/messaging" />
```

3. Replace **false** with **true**. This section will look similar to:

```
<MessageMapper classname="com.ibm.commerce.messaging.programadapter.
  messagemapper.soapecsax.ECSAXMessageMapper"
  enable="true"
  messageMapperId="-3"
  name="WCS.SOAPINTEGRATION">
  <configuration EcInboundMessageDtdFiles=""
    EcInboundMessageDtdPath="WC_installdir/xml/messaging"
    EcSaxParserClass="com.ibm.xml.parsers.SAXParser"
    EcSystemTemplateFile="webservice_template.xml"
    EcTemplatePath="WC_installdir/xml/messaging" />
```

Note: The *EcTemplatePath* and *EcInboundMessageDtdPath* paths should be changed to the actual installation directory. For example, */WebSphere/CommerceServer/xml/messaging*.

4. Locate the following section in the *instance_name.xml* file:

```
HttpAdapter deviceFormatId="-10000"
deviceFormatType="XmlHttp"
deviceFormatTypeId="-10000"
enabled="false"
factoryClassName="com.ibm.commerce.programadapter.HttpProgramAdapterImpl"
name="XML/HTTP">
```

5. Replace **false** with **true**. This section will look similar to:

```
HttpAdapter deviceFormatId="-10000"
deviceFormatType="XmlHttp"
deviceFormatTypeId="-10000"
enabled="true"
factoryClassName="com.ibm.commerce.programadapter.HttpProgramAdapterImpl"
name="XML/HTTP">
```

6. Save and close the *instance_name.xml* file.

Security considerations

The following sections cover the security considerations involved when enabling Web services.

Web services and access control

Access control is the process of restricting access to the protected resources within the WebSphere Commerce system only to authorized individuals and organizations. To facilitate database management and ensure security, WebSphere Commerce has an access control mechanism, which restricts access to resources to specific users. Access control can be defined as security guidelines that:

1. Allow or deny a user of a system access to the resources managed by a system.
2. Specify what actions the user can perform on each resource.

The access control model for WebSphere Commerce is based on the enforcement of access control policies. To learn more about the access control and policies, refer to the *WebSphere Commerce Security Guide*.

When WebSphere Commerce acts as a service provider, it allows external Web service clients to invoke business processes that are defined by controller commands. Consequently, Web services inherit the access control policies that are applicable to the command invoked by the Web service clients. Therefore, credential information needs to be defined for each WSDL definition that is used with WebSphere Commerce. In addition to passing in the credentials required to authenticate the user, the Web service client must ensure that the users have the required authority to execute the commands.

The XML schema format in the WSDL definition for the credentials must look like the following:

```
<complexType
name="com.ibm.commerce.webservice.datatype.Credentials">
<all>
<element name="password" nillable="true" type="string"/>
<element name="loginId" nillable="true" type="string"/>
</all>
</complexType>
```

The credential information needs to be mapped to the following fields in the template definition available in the mapping template file:

- The field name for the user login is 'loginId' and FieldInfo is 'CONTROL'.
- The field name for password is 'loginPassword' and FieldInfo is 'CONTROL'.

Transport level security

To enable transport level security, SOAP messages can be sent over HTTPS. It is strongly recommended that the service provider use HTTPS to send the SOAP messages; otherwise anybody who intercepts the messages can see the credentials that are passed.

Note: To enable HTTPS, the SOAP client must use the URL with HTTPS as the protocol name. Before you invoke the call from the client, set the properties to enable Secure Socket Layer (SSL). If you are using the IBM Developer Kit, Java Technology Edition Version 1.2 or later for the Web service client, then the code to enable HTTPS looks similar to the following:


```
System.setProperty("java.protocol.handler.pkgs",  
com.ibm.net.ssl.internal.www.protocol");
```

```
java.security.Security.addProvider  
(new com.ibm.jsse.JSSEProvider());
```

The preceding code is used if you are using WebSphere Commerce Studio to develop the Web service client.

If you are using the J2SE SDK (Java 2 Standard Edition Software Development Kit) from Sun Microsystems, the client code to invoke the Web service over HTTPS looks similar to the following:

```
System.setProperty("java.protocol.handler.pkgs", "  
com.sun.net.ssl.internal.www.protocol");
```







```
java.security.Security.addProvider  
(newcom.sun.net.ssl.internal.ssl.Provider());
```

If you receive a 'handshake failed', 'unknown CA', or 'connection refused by server exception' at the client side when trying to invoke an inbound service defined by WebSphere Commerce, it means that the client does not recognize the certificate presented by the server. If this is the case and you trust the server, then you can add the server certificate to your list of trusted certificates.

Generating client code to access Web services

This section describes how to generate the client code.

The sample Web services are published using the WSDL definition files and are located in the following directory:

-  *WC_installdir*\samples\webservices\xml\messaging
-  *WCStudio_installdir*\Commerce\samples\webservices\xml\messaging
-    *WC_installdir*/samples/webservices/xml/messaging
-  */QIBM/ProdData/CommerceServer55/samples/webservices/xml/messaging*


You can use the WSDL definition to develop client applications as Web service requestors. You can generate the client code and data types required to access the Web services implemented by WebSphere Commerce from the WSDL definition of the Web service, using an appropriate tool.


IBM WebSphere Studio Application Developer (also referred to as WebSphere Studio in this document) is one of the tools that you can use to generate the client and proxy code from the WSDL definition. For more information, refer to "Appendix B. Using IBM WebSphere Studio Application Developer" on page 47.

Another tool that you can use to generate the client proxy is the Web Services Toolkit. Refer to, <http://www.alphaworks.ibm.com/webservices>.

Sample Web services that can be invoked by external Web clients

This section describes how you can enable the following sample Web services:

1.  OrderCreate service
2. OrderStatus service

You can enable the sample Web services either on a WebSphere Commerce machine or a development machine installed with WebSphere Commerce Studio. If you are enabling the sample Web services on a WebSphere Commerce Studio machine, then follow the instructions given in this section and refer to the directory paths marked with  .

Importing the sample code

If you want to edit or modify the sample Web services, then you must import the sample source code into WebSphere Commerce Studio. To import this code, do the following:

1. Open the WebSphere Commerce development environment from **Start > Programs > IBM WebSphere Commerce Studio > WebSphere Commerce development environment**.
2. Switch to the J2EE perspective and select the J2EE Navigator view.
3. Expand the **WebSphereCommerceServerExtensionsLogic** project. Right-click the **src** directory and select **Import**. The Import Wizard opens.
4. Select **File system** and click **Next**.
5. Click **Browse** and locate the following directory:
`WCStudio_installdir\Commerce\samples\webservices\com`
6. Click **Select All**, then click **Finish**.
7. Right-click the **WebSphereCommerceServerExtensionsLogic** project and select **Properties**.
8. Click **Java Build Path** and then move to the **Libraries** tab.
9. Click **Add variable**. Select the variables, SOAPJAR and XERCES and click **OK**.
10. Click **OK**.
11. Right-click the **WebSphereCommerceServerExtensionsLogic** project and select **Rebuild project**.

You have now completed importing the sample source code into WebSphere Commerce Studio. For more information on how to start the WebSphereCommerceServer server within the WebSphere Test Environment refer to the *WebSphere Commerce Studio Installation Guide*

OrderCreate service (Business Edition)

The OrderCreate Web service applies to WebSphere Commerce Business Edition and WebSphere Commerce Business Developer Edition.



This section demonstrates how to enable the business logic in WebSphere Commerce as a Web service that supports receiving orders from external applications, such as procurement applications. The procurement system can create orders in WebSphere Commerce using the WebSphere Commerce BatchOrderRequest command. This sample OrderCreate Web service implementation explains how you can create a Web service that receives orders from a buyer organization. To use the OrderCreate sample Web service you require a published store within the WebSphere Commerce system

Typically, the seller must know the buyer organization before creating an order in WebSphere Commerce. Accordingly, the buyer organization must be registered with WebSphere Commerce before the buyer can call the OrderCreate Web service.

Enabling the OrderCreate sample Web service

To enable the sample OrderCreate Web service, do the following:







1. Identify the business logic that you want to expose as a Web service. In this case, allow a buyer organization to create a purchase order using an application that is similar to a procurement system.
2. Identify the command that implements this business logic. In this case, it is the BatchOrderRequest command.
3. Identify the mandatory and optional parameters that the BatchOrderRequest command requires. A description of the BatchOrderRequest command is available in the WebSphere Commerce Production and Development online help. For information on the complete list of parameters, refer to the webservice_template.xml file provided in the following directory:

-  `WC_installdir\samples\webservices\xml\messaging`
-  `WCStudio_installdir\Commerce\samples\webservices\xml\messaging`
-    `WC_installdir/samples/webservices/xml/messaging`
-  `/QIBM/ProdData/CommerceServer55/samples/webservices/xml/messaging`

+
+
+
+

Note: Typically, a procurement system uses the BatchOrderRequest command to submit a PurchaseOrder to WebSphere Commerce. For more information about procurement system integration refer to the WebSphere Commerce Production and Development online help.




4. Create the WSDL definition for the OrderCreate Web service. A sample OrderCreate.wsdl definition file is located in the following directory:

-  `WC_installdir\samples\webservices\xml\messaging`
-  `WCStudio_installdir\Commerce\samples\webservices\xml\messaging`
-    `WC_installdir/samples/webservices/xml/messaging`
-  `/QIBM/ProdData/CommerceServer55/samples/webservices/xml/messaging`

You may modify this WSDL definition to meet the definition requirements for the service that you are providing.

The WSDL definition published for the OrderCreate Web service contains the access point, binding template and the datatype schema that are required to invoke this Web service.

5. Map the incoming SOAP request to the command parameters using the mapping template file. A sample mapping template is located in the following directory:

-  `WC_installdir\samples\webservices\xml\messaging\webservice_template.xml`
-  `WCStudio_installdir\Commerce\samples\webservices\xml\messaging\webservice_template.xml`
-    `WC_installdir/samples/webservices/xml/messaging/webservice_template.xml`
-  `/QIBM/ProdData/CommerceServer55/samples/webservices/xml/messaging/webservice_template.xml`

You can customize this file. If you have modified the sample WSDL definition, then you may need to modify this sample mapping template to match the changes in the WSDL definition.







6. To enable this template, which includes the OrderCreate sample Web service, do the following:

- a. Copy the `webservice_template.xml` file into the following directory:

-  `WC_installdir\xml\messaging`
-  `WCStudio_installdir\Commerce\xml\messaging`
-    `WC_installdir/xml/messaging`
-  `/QIBM/UserData/CommerceServer55/
instances/instance_name/xml/messaging`

- b. If the `webservice_template.xml` file already exists and does not contain any information, then overwrite it. If the file contains information, then copy the TemplateDocument information corresponding to the OrderCreate service into the existing file.

7. Enable the message mapper for "SOAPINTEGRATION". To do this open the `instance_name.xml` configuration file located in the following directory:







-  `WC_installdir\instances\instance_name\xml`
-  `WCStudio_installdir\Commerce\instances\instance_name\xml`
-    `WC_installdir/instances/instance_name/xml`
-  `/QIBM/UserData/CommerceServer55/instances/instance_name/xml`

Locate the message mapper with the name "WCS.SOAPINTEGRATION". If the value of the "enable" attribute is "false", then set it to "true".






8. Write JSP templates to compose a response. In this scenario, copy the following sample JSP templates:

- `SOAP_PurchaseOrderResponse.jsp` - this composes the SOAP response message to the OrderCreate request.
- `SOAP_GenericApplicationError.jsp` - this composes the SOAP error response in case of an application error.
- `SOAP_GenericSystemError.jsp` - this composes the SOAP error response in case of a system error.

Copy from the following directory:

-  `WC_installdir\samples\webservices\wcstores`
-  `WCStudio_installdir\Commerce\samples\webservices\wcstores`
-    `WC_installdir/samples/webservices/wcstores`
-  `WC_installdir/samples/webservices/wcstores`

into the following directory:

-  `WAS_installdir\installedApps\cell_name\
WC_instance_name.ear\Stores.war`
where, `cell_name` is the name of the machine where you have installed WebSphere Application Server.
-  `WCStudio_workspacedir\Stores\Web Content`
-    `WAS_installdir/installedApps/cell_name\
WC_instance_name.ear/Stores.war`

- `> 400 WAS_userdir/installedApps/cell_name/
WC_instance_name.ear/Stores.war`
9. Create a buyer organization and a contract between the buyer and supplier organizations. For more information, refer to “Creating users and contracts” on page 20 in this document.
 10. Populate the database tables as described in the “Populating database tables” on page 21 in this document.
 11. Restart the Commerce Server instance for the changes to take effect if you are using WebSphere Commerce. If you are using a WebSphere Commerce Studio machine, then restart the WebSphereCommerceServer server in the WebSphere Test Environment.

Testing the sample

You can generate the Web service client from the WSDL definition. The client can invoke this Web service by sending a PurchaseOrderRequest message. If the order is placed successfully, then the response message contains a status code (200), an “OK” status message, status text, and the OrderID created in WebSphere Commerce. In case of an error you will receive an appropriate error message.

+ Invoking the OrderCreate Web service requires you to fill in many parameters such
 + as the logonId, password, credentials like sender identity, domain and shared
 + secret, orderId, and others. These values will be passed as parameters into the
 + BatchOrderRequest command. Refer to the Web service mapping template to see
 + which element in the SOAP message is mapped to the corresponding parameter in
 + the BatchOrderRequest command.

For more information on testing the Web service, see “Testing the Web service” on page 48 in Appendix B.

Security for the OrderCreate Web service

The business logic that is exposed as a Web service in this sample requires that the requesting buyer organization be authenticated. With the WebSphere Commerce access control model, authentication for commands and their associated Web services can be accomplished by providing the requestor’s WebSphere Commerce user ID and password. This requires that the user be registered as a WebSphere Commerce user.

Authenticating a user, who wants to use the sample OrderCreate Web service, requires the buyer organization to which the user belongs to be registered with WebSphere Commerce. The credentials passed must be verified before processing the OrderCreate service.

The credentials for the OrderCreate Web service include the organization code and the organization code type for the supplier and buyer in the WebSphere Commerce system. The client must send the same value of the organization code and the organization code type as registered in the ORGCODE table.

The following is the schema in the WSDL definition for the credentials, where the identity is the organization code, the domain is the organization code type, and the shared secret is a password that is shared by the buyer and supplier organizations:

```
<complexType
name="com.ibm.commerce.webservice.datatype.ordercreate.
Credential">
<all>
```

```

<element name="identity" nillable="true" type="string"/>
<element name="domain" nillable="true" type="string"/>
<element name="sharedSecret" nillable="true" type="string"/>
</all>

```

Note: The shared secret is applicable only to the sender credentials.

Access Control

When WebSphere Commerce receives the OrderCreate message, it invokes the BatchOrderRequest command to process the incoming request. To use the OrderCreate Web service, a contract must exist between the supplier who is using WebSphere Commerce and the buyer who is acting as the Web service client.

Creating users and contracts

If a buyer organization does not exist, then complete the following before you call the OrderCreate Web service. Using these steps create a new buyer organization, assign the role of a procurement buyer to it and create a contract between the seller and the buyer organizations:

- + 1. Create a buyer business organization using the Organization Administration Console. For details refer to the WebSphere Commerce Production and Development online help.
- + 2. Assign the following roles to this buyer organization:
 - + a. Procurement Buyer
 - + b. Procurement Buyer Administrator
- + 3. Create a user for this buyer organization from the Organization Administration Console. The user ID for this user is the group user ID of the buyer organization and the password must be the shared secret of the buyer organization. The group user ID and shared secret are part of the buyer organization credentials mentioned in the "Security considerations" on page 14. The group user ID and the password are assigned when the buyer organization is registered with the supplier.
- + 4. Assign the following roles to the user created in step 3:
 - + a. Procurement Buyer
 - + b. Procurement Buyer Administrator
- + 5. Create a business account for this buyer organization using the WebSphere Commerce Accelerator. For instructions on how to create a business account refer to the WebSphere Commerce Production and Development online help. When creating a business account note the following:
 - + • Do not select the **Allow customers to purchase under the terms and conditions of store's default contract** check box in the Customer page.
 - + • Do not select the **Purchase order number may be specified at the time of the order** check box in the Purchase Order page.
 - + • Enable the **credit line payment method** and specify the **credit line account number** in the Credit Line page.
- + 6. Create a contract for the business account created. For instructions on how to create a contract for the business account refer to the WebSphere Commerce Production and Development online help. When creating the contract note the following:
 - + a. Click the **General** tab. Enter the Contract name and Description, and select the duration for which you want the contract to be active.
 - + b. Click the **Customer** tab. Select the buyer organization that you created previously.



- + c. Expand Products and Prices in the Contract notebook.
- + 1) Navigate to the Percentage Pricing page.
- + 2) Click **Add**. The Add Percentage Pricing dialog opens.
- + 3) Select **Apply an adjustment on the entire master catalog**.
- + 4) In the Adjustment fields, complete the % field and select **Markdown** from the drop-down list.
- + 5) Click **OK** to close the Add Percentage Pricing dialog.
- d. Click the **Payment** tab. Ensure that **Credit line** usage is checked.
- 7. Click **Submit** to activate the contract.












The preceding steps create an account for the buyer organization in WebSphere Commerce, and a contract between the supplier (WebSphere Commerce) organization and the buyer organization. The necessary access permissions are assigned to the buyer organization.






Populating database tables

To execute the BatchOrderRequest command, you must populate certain tables by executing a database script.

To update your database, do the following:







1. Navigate to the following directory:
 -  `WC_installdir\samples\webservices\bin`
 -  `WCStudio_installdir\Commerce\samples\webservices\bin`
 -     `WC_installdir/samples/webservices/bin`
2. Locate the correct set of script files for your configuration, as described in the following table:

Operating system	Database type	Script file names
 		Webservices_DBUpdate.db2.bat Webservices_UndoDBUpdate.db2.bat
		Webservices_DBUpdate.oracle.bat Webservices_UndoDBUpdate.oracle.bat
 		Webservices_DBUpdate.db2.sh Webservices_UndoDBUpdate.db2.sh
		Webservices_DBUpdate.oracle.sh Webservices_UndoDBUpdate.oracle.sh
 		Webservices_DBUpdate.db2.sh Webservices_UndoDBUpdate.db2.sh





3. Copy the appropriate script files into the following directory:
 -  `WC_installdir\bin`
 -  `WCStudio_installdir\Commerce\bin`
 -    `WC_installdir/bin`

6. After you update the SQL scripts, execute the appropriate command script file:

 If you are using a DB2® database:

-   From a DB2 command window run Webservices_DBUpdate.db2.bat
-    From a command window run Webservices_DBUpdate.db2.sh
-  Start a QSH session. Run the command /QIBM/ProdData/CommerceServer55/samples/webservices/bin/Webservices_DBUpdate.db2.sh

 If you are using an Oracle database:

-   From a command window run Webservices_DBUpdate.oracle.bat
-   From a command window run Webservices_DBUpdate.oracle.sh

Verify the `WC_installdir\logs\Webservices_DBUpdate.log` log file and ensure that the execution completes successfully.







Note: If you want to enable only the OrderCreate Web services, then you must comment out the lines in the SQL script that register the `ExtOrderProcessServiceCmdImpl` and the `WSDLFinderCmdImpl` task command for the OrderFulfillment Web service.

New message mappings







To enable the OrderCreate Web service, a new set of sample mappings is provided with the `webservice_template.xml` file. The following is the template document for the message map:

```
<!-- mapping for inbound order create message -->
<TemplateDocument>
  <DocumentType version='1.0'>ns1:orderCreate</DocumentType>
  <StartElement>ns1:orderCreate</StartElement>
  <TemplateTagName>BatchOrderCreate10Map</TemplateTagName>
  <CommandMapping>
    <Command CommandName='BatchOrderRequest'>
      <Constant Field='protocolName'>SOAP</Constant>
      <Constant Field='protocolVersion'>1.1</Constant>
    </Command>
  </CommandMapping>
</TemplateDocument>
```

The corresponding message mapping for this service is called `BatchOrderCreate10Map` and is available in the `webservice_template.xml` template file, in the following directory:

-  `WC_installdir\samples\webservices\xml\messaging`
-  `WCStudio_installdir\Commerce\samples\webservices\xml\messaging`
-    `WC_installdir/samples/webservices/xml/messaging`
-  `/QIBM/ProdData/CommerceServer55/samples/webservices/xml/messaging`

The message map provided, corresponds to the WSDL definition given in the following directory:

-  `WC_installdir\samples\webservices\xml\messaging\OrderCreate.wsdl`
-  `WCStudio_installdir\Commerce\samples\webservices\xml\messaging\OrderCreate.wsdl`
-    `WC_installdir/samples/webservices/xml/messaging/OrderCreate.wsdl`
-  `/QIBM/ProdData/CommerceServer55/samples/webservices/xml/messaging/OrderCreate.wsdl`

It works with any SOAP request message sent by a Web service client, which is generated from the WSDL definition provided.

Defining the message map requires knowledge of the XPATH information in the SOAP request. You can obtain the XPATH information from the WSDL definition. Alternatively, it may be more convenient to look at the SOAP message. The next section describes how you can access the SOAP message and define a mapping template.

To define a message map similar to the one corresponding to the OrderCreate message (BatchOrderCreate10Map) from the corresponding SOAP request, do the following:

1. Use the WSDL definition to generate the client proxy. Refer to the steps described in “Appendix B. Using IBM WebSphere Studio Application Developer” on page 47.
2. Use a transport level interception utility to see what the SOAP request and SOAP response documents look like. For example, the TCP Monitor available in the WebSphere Studio Application Developer 5.0, or any other HTTP tunneling utility, which can log the message. Allow the client proxy to point to the interceptor by setting the access point. Viewing the SOAP request helps in writing the mappings for the request, and writing the JSP file to build the response.
3. Define the TemplateDocument element of the template definition file. This contains the details of the command that will execute when the SOAP XML message is received. For more information, refer to the section titled, “TemplateDocument element of a template definition file”, in the WebSphere Commerce Production and Development online help.
4. Use the SOAP request that you obtained in step 2 to define the TemplateTag that contains a list of tag definitions. Ensure that the TemplateTag contains mappings for all the elements present in the WSDL definition. This requires that you populate the data for all the elements in the SOAP request when sending the request from the client. Each tag definition maps an element or attribute in the incoming SOAP XML message to a command parameter. Refer to the section titled, “TemplateTag element of a template definition file” in the WebSphere Commerce Production and Development online help.

Configuring message mappers

The SOAP message mapper uses the `webservice_template.xml` Web service template definition file to map the incoming SOAP request to the command properties. As a result, you can configure the SOAP message mapper by changing the template.

If you have a different WSDL definition or want to change the service definition, then you must modify the map in the template. The maps consist of the XPATH

definitions of the XML message mapped to the command parameters. In this case, you need to change the XPATH structure to suit the schema of the datatype according to the new service definition.

OrderStatus service







This section demonstrates how to enable the business logic in WebSphere Commerce that updates the status of an order placed in WebSphere Commerce as a Web service. Using the OrderStatus Web service, you can invoke the WebSphere Commerce OrderStatus command through a Web service call. The OrderStatus controller command allows an external system to update information related to the status of an existing order within WebSphere Commerce. To use the OrderStatus sample Web service you must have an existing order, which requires a published store within the WebSphere Commerce system.

The sample provided shows how to enable the WebSphere Commerce OrderStatus command for Web service requests.







Enabling the OrderStatus sample Web service







The following is an overview of the steps to implement the sample OrderStatus Web service:







1. Identify the business logic that you want to expose as a Web service. In this case, allow an application fulfilling the order to update the status of the order in WebSphere Commerce.
2. Identify the command that implements this business logic. Here, it is the sample OrderStatusService command. It enhances the existing OrderStatus command.
3. Identify the mandatory and optional parameters that this command requires. This information is available in the WebSphere Commerce Production and Development online help.
4. Create the WSDL definition for the OrderStatus Web service. The sample OrderStatus.wsdl definition file is located in the following directory:

-  `WC_installdir\samples\webservices\xml\messaging`
-  `WCStudio_installdir\Commerce\samples\webservices\xml\messaging`
-    `WC_installdir/samples/webservices/xml/messaging`
-  `/QIBM/ProdData/CommerceServer55/samples/webservices/xml/messaging`

5. If you have enabled the OrderCreate service, then do not execute this step. Otherwise, map the incoming SOAP request to the command parameters using the mapping template file. The sample mapping template is located in the following directory:







-  `WC_installdir\samples\webservices\xml\messaging\webservice_template.xml`
-  `WCStudio_installdir\Commerce\samples\webservices\xml\messaging\webservice_template.xml`
-    `WC_installdir/samples/webservices/xml/messaging/webservice_template.xml`
-  `/QIBM/ProdData/CommerceServer55/samples/webservices/xml/messaging/webservice_template.xml`

- a. Copy the `webservice_template.xml` file into the following directory:
 -  `WC_installdir\xml\messaging`
 -  `WCStudio_installdir\Commerce\xml\messaging`
 -    `WC_installdir/xml/messaging`
 -  `/QIBM/UserData/CommerceServer55/instances/
instance_name/xml/messaging`
 - b. If the `webservice_template.xml` file already exists and does not contain any information, then overwrite it. If the file contains information, then copy the `TemplateDocument` corresponding to the `OrderStatus` service into the existing file.
6. If you have enabled the `OrderCreate` Service, then do not execute this step. Otherwise, enable the message mapper for "SOAPINTEGRATION". To do this open the `instance_name.xml` configuration file located in the following directory:







-  `WC_installdir\instances\instance_name\xml`
-  `WCStudio_installdir\Commerce\instances\instance_name\xml`
-    `WC_installdir/instances/instance_name/xml`
-  `/QIBM/UserData/CommerceServer55/instances/instance_name/xml`

Locate the message mapper with the name "WCS.SOAPINTEGRATION". If the value of the "enable" attribute is "false", then set it to "true".

7. Write JSP templates to compose a response. The sample JSP file `SOAP_OrderStatusResponse.jsp` is available to compose a response.
 - a. Copy the `SOAP_OrderStatusResponse.jsp` from the following directory:



-  `WC_installdir\samples\webservices\wcstores`
-  `WCStudio_installdir\Commerce\samples
\webservices\wcstores`
-    `WC_installdir/samples/webservices/wcstores`
-  `WC_installdir/samples/webservices/wcstores`




into the following directory:

-  `WAS_installdir\installedApps\cell_name\
WC_instance_name.ear\Stores.war`
-  `WCStudio_installdir\Commerce\IBM WebSphere Test
Environment\hosts\default_host\default_app\web`
-    `WAS_installdir/installedApps/cell_name/
WC_instance_name.ear/Stores.war`
-  `WAS_userdir/installedApps/cell_name/
WC_instance_name.ear/Stores.war`






This JSP template composes the SOAP response message when the status of the order is updated successfully in WebSphere Commerce.

- b. If you have enabled the `OrderCreate` service, then you do not need to execute this step. Otherwise, copy the `SOAP_GenericApplicationError.jsp`, and `SOAP_GenericSystemError.jsp` files from the following directory:

-  `WC_installdir\samples\webservices\wcstores`
-  `WCStudio_installdir\Commerce\samples
\webservices\wcstores`

-    *WC_installdir/samples/webservices/wcstores*
-  */QIBM/ProdData/CommerceServer55/samples/webservices/wcstores*







into the following directory:

-  *WAS_installdir\installedApps\cell_name\WC_instance_name.ear\Stores.war*
-  *WCStudio_workspacedir\Stores\Web Content*
-    *WAS_installdir/installedApps/cell_name/WC_instance_name.ear/Stores.war*
-  *WAS_userdir/installedApps/cell_name/WC_instance_name.ear/Stores.war*












These JSP templates compose the SOAP error messages in case of an application or a system error.

8. If you have enabled the OrderCreate service, then you do not need to execute this step. Otherwise, do the following:







a. Navigate to the following directory:








-  *WC_installdir\samples\webservices\bin*
-  *WCStudio_installdir\Commerce\samples\webservices\bin*
-     *WC_installdir/samples/webservices/bin*

b. Locate the correct set of script files for your configuration, as described in the following table:







Operating system	Database type	Script file names
 		Webservices_DBUpdate.db2.bat Webservices_UndoDBupdate.db2.bat
		Webservices_DBUpdate.oracle.bat Webservices_UndoDBupdate.oracle.bat
 		Webservices_DBUpdate.db2.sh Webservices_UndoDBupdate.db2.sh
		Webservices_DBUpdate.oracle.sh Webservices_UndoDBupdate.oracle.sh
 		Webservices_DBUpdate.db2.sh Webservices_UndoDBupdate.db2.sh


c. Copy the appropriate script files into the following directory:





-  *WC_installdir\bin*
-  *WCStudio_installdir\Commerce\bin*
-    *WC_installdir/bin*
-  */QIBM/ProdData/CommerceServer55/bin*

- d.  Copy the `wcs.webservices.enable.sql` and `wcs.webservices.disable.sql` scripts from the following directory:
-  `WC_installdir\samples\webservices\schema\db2`
 -  `WCStudio_installdir\Commerce\samples\webservices\schema\db2`
 -    `WC_installdir/samples/webservices/schema/db2`
 -  `/QIBM/ProdData/CommerceServer55/samples/webservices/schema/db2`





into the following directory:

-  `WC_installdir\schema\db2`
-  `WCStudio_installdir\Commerce\schema\db2`
-    `WC_installdir/schema/db2`
-  `/QIBM/ProdData/CommerceServer55/schema/db2`

 Copy the `wcs.webservices.enable.sql` and `wcs.webservices.disable.sql` scripts from the following directory:

-  `WC_installdir\samples\webservices\schema\oracle`
-  `WCStudio_installdir\Commerce\samples\webservices\schema\oracle`
-   `WC_installdir/samples/webservices/schema/oracle`

into the following directory:

-  `WC_installdir\schema\oracle`
-  `WCStudio_installdir\Commerce\schema\oracle`
-   `WC_installdir/schema/oracle`

- e. Update the SQL script files that you have copied in step 7c by commenting out the script statements that are not related to the OrderStatus Web service.

- f. Execute the script files that you have copied in step 7b.

9. Restart the Commerce Server instance for the changes to take effect if you are using a WebSphere Commerce machine. If you are using a WebSphere Commerce Studio machine, then restart the WebSphereCommerceServer server in the WebSphere Test Environment.

Testing the sample

You can generate the Web service client from the WSDL definition. The client can invoke this Web service by sending an OrderStatusUpdate message. If the order status is updated successfully, then the response message contains a status code (200), an "OK" status message, status text, and the row ID of the order status record created in WebSphere Commerce. In case of an error you will receive an appropriate error message.

Invoking the OrderStatus Web service requires you to fill in many parameters like the `logonId`, `password`, `orderNumberByWC`, `orderNumberByMerchant`, and others. These values will be passed as parameters into the `OrderStatusService` command. The Web services mapping template is used to map the elements in the inbound SOAP message to a command parameter.

+
+

For more information on testing the Web service, see “Testing the Web service” on page 48 in Appendix B.

Access control for the OrderStatus service

For the OrderStatus Web service, the WebSphere Commerce access control model is used. This requires a User ID and password to be passed on the Web service request so that WebSphere Commerce can authenticate the requestor and verify that the requestor has authority to perform the required action. The User ID and password are defined in the WSDL definition as follows:

The XML schema format for the credentials must look like the following:

```
<complexType
name="com.ibm.commerce.webservice.datatype.Credentials">
<all>
<element name="password" nillable="true" type="string"/>
<element name="logonId" nillable="true" type="string"/>
</all>
</complexType>
```

The credential information needs to be mapped to the following fields in the template definition:

- The field name for the user login is 'logonId' and FieldInfo is 'CONTROL'.
- The field name for password is 'logonPassword' and FieldInfo is 'CONTROL'.

For example, for the OrderStatus service the LogonID and password are mapped to the OrderStatus command using the following section in the mapping template:

```
<TemplateTag name='OrderStatus20Map'>
<!--Execution Environment -->
<Tag XPath='updateOrderStatusMessage/controlArea/credentials/logonId'
Field='logonId' FieldInfo='CONTROL' />
<Tag XPath='updateOrderStatusMessage/controlArea/credentials/password'
Field='logonPassword' FieldInfo='CONTROL' />
```






This sample provides a typical method to meet access control requirements for Web services that are defined for WebSphere Commerce.


User roles for the OrderStatus Web service

The OrderStatus service uses the OrderStatusServiceCmd and theOrderStatusCmd controller commands. By default, these controller commands are added as resources into the SellersCmdResourceGroup resource group and as actions into the BackendOrderStatusCreate action group. As a result, only users assigned with the role of a 'seller' can execute the OrderStatusCmd and OrderStatusServiceCmd commands. This means that the user whose credentials are passed in as a part of the OrderStatus message must have the 'seller' role. (By default, any user with the WebSphere Commerce administrator authority can execute any of these commands. Consequently, this is applicable only if you want to enable the OrderStatus service for a user who does not have administrator authority

The OrderStatus WSDL definition

The OrderStatus Web service is defined according to the OrderStatus WSDL definition available in the following directory:

-  `WC_installdir\samples\webservices\xml\messaging`
-  `WCStudio_installdir\Commerce\samples\webservices\xml\messaging`
-    `WC_installdir/samples/webservices/xml/messaging`







-  /QIBM/ProdData/CommerceServer55/samples/webservices/xml/messaging

New message mappings

To enable the OrderStatus Web service, a new set of mappings is provided in the mapping template `webservice_template.xml` file. The following is the template document for the message map:

```
<!-- mapping for inbound order status update message -->
<TemplateDocument>
  <DocumentType version='1.0'>ns1:update_WCS_OrderStatus
  </DocumentType>
  <StartElement>ns1:update_WCS_OrderStatus</StartElement>
  <TemplateTagName>OrderStatus20Map</TemplateTagName>
  <CommandMapping>
    <Command CommandName='OrderStatusService'>
      Condition='OrderStatusCommand'>
      <Constant Field='redirecturl'>
        SOAP_OrderStatusResponse
      </Constant>
      <Constant Field='SOAP_OrderStatusResponse'>
        SOAP_OrderStatusResponse.jsp
      </Constant>
    </Command>
  </CommandMapping>
</TemplateDocument>
```

The corresponding message mapping for this service is called `OrderStatus20Map` and is available in the `webservice_template.xml` template file, in the following directory:

-  `WC_installdir\samples\webservices\xml\messaging`
-  `WCStudio_installdir\Commerce\samples\webservices\xml\messaging`
-    `WC_installdir/samples/webservices/xml/messaging`
-  /QIBM/ProdData/CommerceServer55/samples/webservices/xml/messaging

The message map provided, corresponds to the WSDL definition described in the following directory:

-  `WC_installdir\samples\webservices\xml\messaging\Orderstatus.wsdl`
-  `WCStudio_installdir\Commerce\samples\webservices\xml\messaging\Orderstatus.wsdl`
-    `WC_installdir/samples/webservices/xml/messaging/Orderstatus.wsdl`
-  /QIBM/ProdData/CommerceServer55/samples/webservices/xml/messaging/OrderStatus.wsdl

The message map works with any message sent by a Web service client that is generated from the WSDL definition provided.

Defining the message map requires knowledge of the XPATH information in the SOAP request. You can obtain the XPATH information from the WSDL definition. Alternatively, it may be more convenient to look at the SOAP message. The next section describes how you can access the SOAP message and define a mapping template.

Do the following to define a message map similar to the one corresponding to the OrderStatus message (OrderStatus20Map) from the corresponding SOAP request:

1. Use the WSDL definition file to generate the client proxy. Refer to the steps described in “Appendix B. Using IBM WebSphere Studio Application Developer” on page 47 for more information.
2. Use a transport level interception utility to see what the SOAP request and SOAP response documents look like. For example, the TCP Monitor available in the Apache AXIS distribution, or any other HTTP tunneling utility, which can log the message. Allow the client proxy to point to the interceptor by setting the access point. This helps in writing the mappings for the request, and writing the JSP file to build the response.
3. Define the TemplateDocument element of the template definition file. This contains the details of the command that will execute when an XML message is received. For more information, refer to the section titled, “TemplateDocument element of a template definition file”, in the WebSphere Commerce Production and Development online help.
4. Define the TemplateTag that contains a list of tag definitions. Ensure that the TemplateTag contains mappings for all the elements present in the WSDL definition. This requires that you populate the data for all the elements in the SOAP request when sending the request from the client. Each tag definition maps an element or attribute in the incoming XML message to a command parameter. Refer to the section titled, “TemplateTag element of a template definition file” in the WebSphere Commerce Production and Development online help.

Configuring message mappers

The SOAP message mapper uses the Web service template definition file (webservice_template.xml) to map the incoming SOAP request to the command properties. As a result you can configure the SOAP message mapper by modifying the template. For more information about the inbound message template definition files, refer to the WebSphere Commerce Production and Development online help.

If you have a different WSDL definition or want to change the service definition, then you must modify the map in the template. The maps consist of the XPATH definitions of the XML message mapped to the command parameters. In this case, you need to change the XPATH structure to suit the schema of the datatype according to the new service definition.



Disabling the OrderCreate and OrderStatus Web services





This section explains how to disable the OrderCreate and OrderStatus Web services provided by WebSphere Commerce.

1. Delete the following JSP files:


- SOAP_PurchaseOrderResponse.jsp
- SOAP_GenericApplicationError.jsp
- SOAP_GenericSystemError.jsp
- SOAP_OrderStatusResponse.jsp

from the following directory:

-  WAS_installdir\installedApps\cell_name\WC_instance_name.ear\Stores.war
-  WCStudio_workspacedir\Stores\Web Content

-    `WAS_installdir/installedApps/cell_name/WC_instance_name.ear/Stores.war`
-  `WAS_userdir/installedApps/cell_name/WC_instance_name.ear/Stores.war`








2. Replace the `webservice_template.xml` file from the following directory:

-  `WC_installdir\xml\messaging`
-  `WCStudio_installdir\Commerce\xml\messaging`
-    `WC_installdir/xml/messaging`
-  `/QIBM/UserData/CommerceServer55/instances/instance_name/xml/messaging`





with an empty template that contains an empty `ECTemplate` node as showing in the following example:

```
<?xml version='1.0' encoding='UTF-8'?>
<!DOCTYPE ECTemplate SYSTEM 'ec_template.dtd'>
<ECTemplate>
</ECTemplate>
```

3. Locate the `wcs.webservices.disable.sql` scripts in the following directory:

- ▶  If you are using a DB2 database:
-  `WC_installdir\schema\db2`
-  `WCStudio_installdir\Commerce\schema\db2`
-    `WC_installdir/schema/db2`
-  `QIBM/UserData/CommerceServer55/instances/instance_name/schema`







▶  If you are using an Oracle database:

-  `WC_installdir\schema\oracle`
-  `WCStudio_installdir\Commerce\schema\oracle`
-   `WC_installdir/schema/oracle`


4. Open the `wcs.webservices.disable.sql` script that you located in step 3, in a text editor, and ensure that the following values are the same as in the `wcs.webservices.enable.sql` script:

- `storeent_id` - is the unique identifier for your store.
- `OrgCode1` - this is the primary key for the entry in the `ORGCODE` table corresponding to the supplier organization.
- `OrgCode2` - this is the primary key for the entry in the `ORGCODE` table corresponding to the buyer organization.
- `MemberId` - this is the primary key of the member group to which the buyer organization belongs.





5. After completing the changes mentioned in step 4 run the appropriate command script file:

- ▶  If you are using a DB2 database:
-   From a DB2 command window run `Webservices_UndoDBUpdate.db2.bat`
-    From a command window run `Webservices_UndoDBUpdate.db2.sh`

+
+

-  400 Start a QSH session. Run the command
/QIBM/ProdData/CommerceServer55/bin/Webservices_UndoDBUpdate.db2.sh

 Oracle If you are using an Oracle database:


-   From a DB2 command window run
Webservices_UndoDBUpdate.oracle.bat
-   From a command window run
Webservices_UndoDBUpdate.oracle.sh

Verify the log file and ensure that the execution completes successfully. The name and location of the log file displays after the execution of the script file is complete.

Note: If you want to disable only the OrderCreate and OrderStatus Web services or you have not enabled the OrderFulfillment Web service, then comment out the following SQL statements from the script files listed in step 4:

- delete from cmdreg where interfacename=
'com.ibm.commerce.webservice.utils.WSDLFinderCmd'
and classname='com.ibm.commerce.webservice.utils.WSDLFinderCmdImpl';
- delete from cmdreg WHERE storeent_id=
storeent_id AND interfacename='
com.ibm.commerce.order.commands.ExtOrderProcessCmd';

where, *storeent_id* is the unique identifier for your store.

6.  Delete the sample code that you imported in “Importing the sample code” on page 16 and then rebuild the WebSphereCommerceServerExtensionsLogic project.
7. Restart the Commerce Server instance for the changes to take effect if you are using a WebSphere Commerce machine. If you are using a WebSphere Commerce Studio machine, then restart the WebSphereCommerceServer server in the WebSphere Test Environment.

Chapter 4. Enabling WebSphere Commerce as a service requestor

This section describes how you can enable WebSphere Commerce to invoke a Web service defined by an external system. It describes the sample provided, which enables WebSphere Commerce to find and invoke Web services. Additionally, the implementation details of the OrderFulfillment sample Web service are explained.

Architecture

The message flow in this architecture is synchronous. This means that WebSphere Commerce waits for a response from the external system before continuing with further processing. The task command is implemented to interact with an external system by calling a Web service.

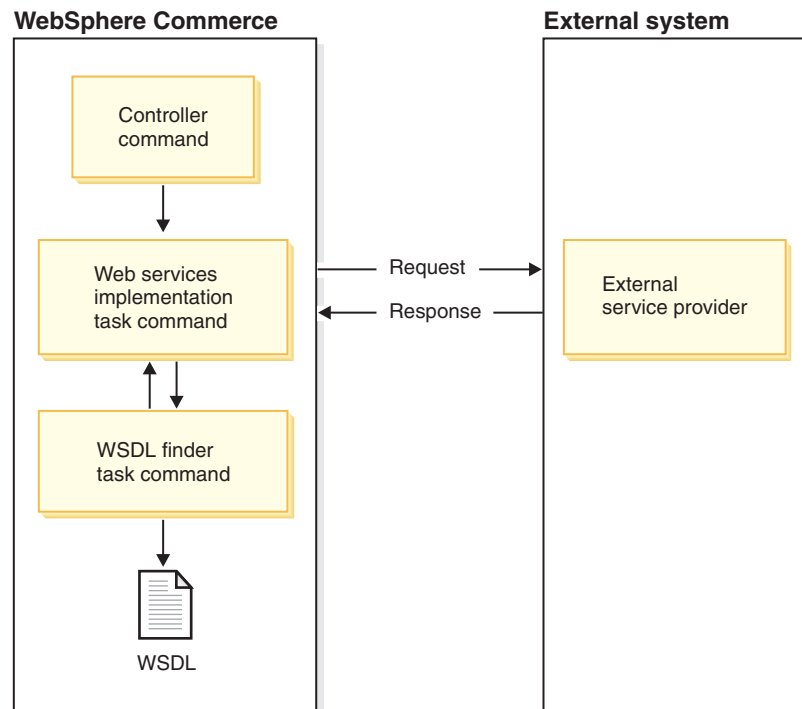


Figure 5. Architecture to enable WebSphere Commerce as a service requestor

The WebSphere Commerce programming model allows you to customize the behavior of an existing business process by overriding commands. A controller command invokes many task commands to complete the execution of a business process. One way to modify the behavior of a controller command is to write a new implementation of a task command that the controller command invokes.

To find out which task commands are invoked as part of a controller command, refer to the section titled "Cross reference of commands, tasks and tables" in the WebSphere Commerce Production and Development online help. The following list describes the interaction when a WebSphere Commerce business process invokes a Web service hosted by an external system:







1. The controller command invokes the registered task command. In this case the task is implemented to interact with an external system by calling a Web service. This task command contains the code to integrate with an external system.
2. The task command retrieves the access point for the specified Web service.
3. The task command invokes a Web service proxy specifying the service access point and its input parameters.
4. The proxy calls the Web service, receives the response and returns it back to the task command.

Finding a service implementation

In order to enable WebSphere Commerce to invoke a Web service defined by an external system, you must first locate a service provider hosting the implementation for the required service type. To invoke a Web service you can point the task command to the identified service provider. For more flexibility a task command is included to locate a service provider, which is defined in the WSDLFinderCmd task command. A sample implementation is provided for this task command and is located in the following directory:

-  `WC_installdir\samples\webservices\com\ibm\commerce\webservice\utils`
-  `WCStudio_installdir\Commerce\samples\webservices\com\ibm\commerce\webservice\utils`
-    `WC_installdir/samples/webservices/com/ibm/commerce/webservice/utils`
-  `/QIBM/ProdData/CommerceServer55/samples/webservices/com/ibm/commerce/webservice/utils`

One way to look for a service implementation is to look into a service registry. However, the sample provided with WebSphere Commerce Version 5.5 uses the local file system to retrieve the WSDL definition of the Web service. The WSDLFinderCmdImpl looks for the registered WSDL definition files available locally in the `instance_name.xml` configuration file. To register a WSDL service definition, edit the `instance_name.xml` configuration file located in the following directory:

-  `WC_installdir\instances\instance_name.xml`
-  `WCStudio_installdir\Commerce\instances\instance_name.xml`
-    `WC_installdir/instances/instance_name.xml`
-  `/QIBM/UserData/CommerceServer55/instances/instance_name.xml`

and do the following:

1. Locate the SOAPWSDLFinderInfo element. This element has attributes to register the WSDL definition path and the WSDL file. The following excerpt from the `instance_name.xml` file shows these parameters:

```
<SOAPWSDLFinderInfo
  LocalWSDLFiles="OrderFulfillment.wsdl"
  LocalWSDLPath="WC_installdir/xml/messaging"
  UDDIQueryURL=""
  UDDIPublishURL=""
  WSDLFinderImplClassName=""
  display="false" />
```

2. Set the value of the LocalWSDLFiles to include the list of comma-separated WSDL definitions that you have on your local file system and the LocalWSDLPath to the directory where these files exist.
3. Set the value of UDDIQueryURL and UDDIPublishURL to an empty string because this sample uses the local file system to look for a service definition.
4. Set the value of WSDLFinderImplClassName to an empty string.

To look for a partner dynamically and invoke a service that the partner has implemented and published in the service registry, you must create and register a new implementation of the WSDLFinderCmdImpl sample task command that is provided with WebSphere Commerce version 5.5. In your custom implementation, include the code to look up the WSDL definition. Here, you can use the UDDIQueryURL and UDDIPublishURL fields to register a UDDI registry. For more information on how to override task commands refer to the *WebSphere Commerce Programming Guide and Tutorials*.

Invoking a service implementation

To find out which task commands are invoked as part of a controller command, refer to the section titled "Cross reference of commands, tasks and tables" in the WebSphere Commerce Production and Development online help.


The following table shows the tasks that are executed as a part of the OrderProcessCmd:

Task command name	Description
CheckOrderCmd	The CheckOrderCmd task command checks if an order, which is in a "submitted" state has been processed.
DoPaymentCmd	The OrderProcessCmd controller command calls the DoPaymentCmd task command to process payment for an order.
ExtOrderProcessCmd	Perform any additional processing required just prior to the completion of the OrderProcess command.
UpdateInventoryCmd	The UpdateInventory task command allows you to update the inventory for items.

From this table it is clear that if you want to customize the behavior of the OrderProcess command to send a fulfillment request to an external fulfillment center, you can override the ExtOrderProcessCmd task command. The new implementation of the ExtOrderProcessCmd can lookup a partner who has implemented the OrderFulfillment service and invoke that service.

Sample Web service that WebSphere Commerce can invoke on an external system

This section describes how you can enable the OrderFulfillment sample Web service.

You can enable the OrderFulfillment sample either on a WebSphere Commerce machine or a development machine installed with WebSphere Commerce Studio. If you are enabling this sample on a WebSphere Commerce Studio machine, then follow the instructions given in this section and refer to the directory paths marked with " Studio".

OrderFulfillment service

This section explains how you can enable WebSphere Commerce to call a Web service implemented by an external system. The OrderFulfillment service is used to demonstrate how WebSphere Commerce can act as a service requestor. To use the OrderFulfillment sample Web service you require a published store within the WebSphere Commerce system.

If you are enabling this sample Web service in the WebSphere Commerce environment, you will also require a WebSphere Commerce Studio environment.

Interface WSDL definitions

The business logic for the OrderFulfillment service is designed to work with a predefined service interface, which is the OrderFulfillment service. The interface definition WSDL file for the OrderFulfillment service is located in the local file system for reference and for default static binding to a known service.

Task commands

The ExtOrderProcessServiceCmdImpl command that is invoked as part of the OrderProcessCmd implements the ExtOrderProcessServiceCmd interface to interact with an external service. It initiates a request with the Web service through the proxy. This task command processes a submitted order by sending an OrderFulfillment request to the respective fulfillment center.

The following are the steps to enable the OrderFulfillment service:

1. If you have already enabled WebSphere Commerce as a service provider as described in Chapter 3, "Populating database tables" on page 21, then do not execute this step. However, if you have commented out the section for the OrderFulfillment commands previously, then you must execute this step.

To register the ExtOrderProcessServiceCmdImpl and the WSDLFinderCmdImpl in the WebSphere Commerce Server command registry, execute the SQL statements related to these commands. The SQL script file is available in the following directory:

▶ **DB2** If you are using a DB2 database:

- **Windows** `WC_installdir\samples\webservices\schema\db2\wcs.webservices.enable.sql`
- **Studio** `WCStudio_installdir\Commerce\samples\webservices\schema\db2\wcs.webservices.enable.sql`
- **AIX** ▶ **Linux** ▶ **Solaris** `WC_installdir/samples/webservices/schema/db2/wcs.webservices.enable.sql`
- **400** `/QIBM/ProdData/CommerceServer55/samples/webservices/schema/db2/wcs.webservices.enable.sql`

▶ **Oracle** If you are using an Oracle database:

- **Windows** `WC_installdir\samples\webservices\schema\oracle\wcs.webservices.enable.sql`
- **Studio** `WCStudio_installdir\Commerce\samples\webservices\schema\oracle\wcs.webservices.enable.sql`
- **AIX** ▶ **Solaris** `WC_installdir/samples/webservices/schema/oracle/wcs.webservices.enable.sql`

The following are the SQL statements from the preceding script file that you must run:

```
insert into cmdreg (storeent_id, interfacename, description,
  classname, properties, lastupdate, target) values(0, 'com.ibm.commerce.
webservice.utils.WSDLFinderCmd', 'Wsd1 finder class for webservices',
com.ibm.commerce.webservice.utils.WSDLFinderCmdImpl',
null, null, 'Local');
```







```
insert into cmdreg (storeent_id, interfacename, description,
  classname, properties, lastupdate, target) values(storeent_id,
'com.ibm.commerce.order.commands.ExtOrderProcessCmd', '
webservices implementation of ExtOrderProcessCmd',
'com.ibm.commerce.webservice.order.commands.ExtOrderProcessServiceCmdImpl',
null,null,'Local');
```

where, *storeent_id* is the unique identifier for your store.






2. To enable lookup for the OrderFulfillment service do the following:
 - a. Locate the SOAPWSDLFinderInfo element in the *instance_name.xml* configuration file.
 - b. Ensure that the values of LocalWSDLFiles and LocalWSDLPath are correct and reflect the directory where the WSDL definition files exist. The SOAPWSDLFinderInfo element must look like the following:

```
<SOAPWSDLFinderInfo
  LocalWSDLFiles="OrderFulfillment.wsdl"
  LocalWSDLPath="WC_installdir/xml/messaging"
  WSDLFinderImplClassName=""
  display="false" />
```

- c. Copy the sample OrderFulfillment.wsdl WSDL file located in the following directory:

-  *WC_installdir\samples\webservices\xml\messaging*
-  *WCStudio_installdir\Commerce\samples\webservices\xml\messaging*
-    *WC_installdir/samples/webservices/xml/messaging*
-  */QIBM/ProdData/CommerceServer55/samples/webservices/xml/messaging*

into the following directory:

-  *WC_installdir\xml\messaging*
-  *WCStudio_Installdir\Commerce\xml\messaging*
-    *WC_installdir/xml/messaging*
-  */QIBM/UserData/CommerceServer55/instances/instance_name/xml/messaging*

3. Identify a service provider for the services defined in the OrderFulfillment.wsdl file. Update the OrderFulfillment.wsdl with the information of the service provider.
 - a. Open the OrderFulfillment.wsdl file that you copied in step 2c.
 - b. Locate the service definition node (the section that begins with "service name"). It looks similar to the following:

```

<service name="POCreate">
<port binding="tns:POCreateBinding" name="POCreatePort">
<soap:address location="http://localhost:8080/WebservicesForWCBEWeb/
  servlet/rpcrouter"/>
</port>
</service>

```

- c. Change the existing highlighted URL to the URL of the target Web service.
 - d. Save and close the file.
4. If you have enabled the OrderCreate or the OrderStatus sample Web service, then ignore steps 4 and 5. Otherwise, proceed with this step. To deploy the sample code provided for the OrderFulfillment Web service, you must have a development machine with WebSphere Commerce Studio installed and configured. The steps in the following list that require WebSphere Studio Application Developer are intended to be performed on this development machine.
 5. If you want to edit or modify the sample Web services, then you must import the sample source code into WebSphere Commerce Studio. To import this code, do the following:
 - a. Open the WebSphere Commerce development environment from **Start > Programs > IBM WebSphere Commerce Studio > WebSphere Commerce development environment**.
 - b. Switch to the J2EE perspective and select the J2EE Navigator view.
 - c. Expand the **WebSphereCommerceServerExtensionsLogic** project. Right-click the **src** directory and select **Import**. The Import Wizard opens.
 - d. Select **File system** and click **Next**.
 - e. Click **Browse** and locate the following directory:
`WCStudio_installdir\Commerce\samples\webservices\com`
 - f. Click **Select All**, then click **Finish**.
 - g. Right-click the **WebSphereCommerceServerExtensionsLogic** project and select **Properties**.
 - h. Click **Java Build Path** and then move to the **Libraries** tab.
 - i. Click **Add variable**. Select the variables, SOAPJAR and XERCES and click **OK**.
 - j. Click **OK**.
 - k. Right-click the **WebSphereCommerceServerExtensionsLogic** project and select **Rebuild project**.
 6. If you are enabling the sample Web service on a WebSphere Commerce server, then you must deploy this code to your target WebSphere Commerce server. Refer to the *WebSphere Commerce Programming Guide and Tutorials* for more information about how to deploy new commands.
 7. Restart the Commerce Server instance for the changes to take effect if you are using a WebSphere Commerce machine. If you are using a WebSphere Commerce Studio machine, then restart the WebSphereCommerceServer server in the WebSphere Test Environment.

Business objects

The ExtOrderProcessServiceCmdImpl task command populates a set of business objects that the Web service proxy uses to construct the SOAP XML request message. This request message reflects the elements of the OrderFulfillment WSDL definition. The OrderFulfillment service comes with a set of sample business objects that are generated from the OrderFulfillment WSDL definition. When WebSphere Commerce invokes external Web services, you can generate the

client proxy and business objects for them using IBM WebSphere Studio, as described in “Appendix B. Using IBM WebSphere Studio Application Developer” on page 47, or using other tools.

The following table lists the packages and their business objects used by the OrderFulfillment service:

Package name	Object name	Description
com.ibm.commerce.webservice.datatype	Address ContactInfo ContactPersonName ControlArea Credentials Email InvoiceInfo ProductDimension ProductMeasurement ProductWeight RequisitionerID Telephone UserData UserDataField	This package consists of all the objects that are used across different Web service messages.


Package name	Object name	Description
com.ibm.commerce.webservice.datatype.orderfulfillment	BillToInfo	This package contains all the objects that contain data specific to the OrderFulfillment XML message. Report_NC_PurchaseOrder is the parent wrapper object that is sent to the proxy and OrderFulfillmentStatus object received as the response.
	BuyOrgAccountingDetail	
	BuyOrgInfo	
	DataArea	
	DateTimeReference	
	ItemShippingSchedule	
	ItemUnitPrice	
	MerchantInfo	
	MonetaryAmount	
	OrderFulfillmentStatus	
	PCardInfo	
	Report_NC_PurchaseOrder	
	ReportPO	
	ReportPOHeader	
	ReportPOItem	
	RequisitionerInfo	
	ServiceAllowanceCharge	
	ShipDateReference	
	ShippingCarrierInfo	
	ShipToInfo	
TaxInfo		
TotalPriceInfo		

Testing the sample

When you place an order in a published store in WebSphere Commerce, the OrderFulfillment Web service is invoked. The Web service connects to the external service provider to place an order with the fulfillment center. If the service provider returns the ID of the order placed, then the OrderID is updated in the ORMORDER column of the ORDERS table in the WebSphere Commerce database.

Disabling the OrderFulfillment Web service

This section explains how to disable the sample OrderFulfillment Web service provided by WebSphere Commerce.

1. Delete the webservices.jar JAR file that you deployed to enable the OrderFulfillment Web service.
2. Delete the OrderFulfillment.wsdl file from the following directory:
 -  `WC_installdir\xml\messaging`

- Studio WCStudio_installdir\Commerce\xml\messaging
- AIX Linux Solaris WC_installdir/xml/messaging
- 400 /QIBM/UserData/CommerceServer55/
instances/*instance_name*/xml/messaging

3. Remove the references to OrderFulfillment.wsdl file from the *instance_name.xml* configuration file.
4. If you have not executed SQL script to disable the OrderCreate and OrderStatus Web services, or commented out the following SQL statements when disabling the OrderCreate and OrderStatus Web services, then execute the following SQL statements:

```
delete from cmdreg where interfacename='com.ibm.commerce.webservice.utils.  
WSDLFinderCmd' and classname='com.ibm.commerce.webservice.utils.  
WSDLFinderCmdImpl';
```

```
delete from cmdreg where storeent_id= storeent_id  
and interfacename='com.ibm.commerce.order.commands.  
ExtOrderProcessCmd';
```

where, *storeent_id* is the unique identifier for your store.

The preceding SQL statements are available in the following script file:

DB2 If you are using a DB2 database:

- Windows Studio From a DB2 command window run
Webservices_UndoDBUpdate.db2.bat
- AIX Linux Solaris From a command window run
Webservices_UndoDBUpdate.db2.sh
- 400 Start a QSH session. Run the Webservices_UndoDBUpdate.db2.sh
command.

Oracle If you are using an Oracle database:







- Windows Studio From a DB2 command window run
Webservices_UndoDBUpdate.oracle.bat
- AIX Solaris From a command window run
Webservices_UndoDBUpdate.oracle.sh

Verify the log file and ensure that the execution completes successfully. The name and location of the log file displays after the execution of the script file is complete.

5. Studio Delete the sample code that you imported in “Importing the sample code” on page 16 and then rebuild the WebSphereCommerceServerExtensionsLogic project.
6. Restart the Commerce Server instance for the changes to take effect if you are using a WebSphere Commerce machine. If you are using a WebSphere Commerce Studio machine, then restart the WebSphereCommerceServer server in the WebSphere Test Environment.

Appendix A. Sample WSDL definition files

The WSDL definitions for the sample Web services provided with WebSphere Commerce Version 5.5 are available in the following directory:

-  *WC_installdir*\samples\webservices\xml\messaging
-  *WCStudio_installdir*\Commerce\samples\webservices\xml\messaging
-  *WC_installdir*/samples/webservices/xml/messaging
-   *WC_installdir*/samples/webservices/xml/messaging
-  /QIBM/ProdData/CommerceServer55/
samples/webservices/xml/messaging

Web service name	File name
OrderFulfillment	OrderFulfillment.wsdl
OrderCreate	OrderCreate.wsdl
OrderStatus	OrderStatus.wsdl

Appendix B. Using IBM WebSphere Studio Application Developer

Various tools can be used to generate the Web service code. The sample code provided with this implementation of Web services is generated using the IBM WebSphere Studio Application Developer. The information in this section was tested on IBM WebSphere Studio Application Developer, Version 5.0. If you do not have WebSphere Commerce or if you are not using the WebSphere Commerce Studio, but want to develop a Web service client to access the Web services provided by WebSphere Commerce, then you can use any tool that is capable of generating the Web service client code.

This appendix explains how to use WebSphere Studio to generate a test client for the services provided by WebSphere Commerce. This involves importing the WSDL definition files into WebSphere Studio, generating a test client, running the test client, creating and configuring a TCP Monitor server instance and using the TCP Monitor to trace the SOAP messages.

You can also use the Web services Toolkit (WSTK) to generate the client code and datatypes. For more information about the Web Services Toolkit, see <http://www.alphaworks.ibm.com/tech/webservicestoolkit>. The WSTK showcases emerging technology in Web services and assists in understanding the latest Web services specifications. However, for a product level development environment of Web services, IBM WebSphere Studio Application Developer is recommended.

Alternatively, you can write the client code and datatypes manually using the Apache SOAP implementation as a client. For more details, refer to the Apache SOAP documentation.







This section assumes knowledge of WebSphere Studio Application Developer. If you are not familiar with developing Web services and enterprise applications with WebSphere Studio, it is highly recommended that you first read the WebSphere Studio documentation or the tutorials provided in the IBM developerworks site (<http://www.ibm.com/developerworks/webservices>).

Importing a WSDL into WebSphere Studio Application Developer

Before importing a WSDL into WebSphere Studio, ensure that a Web application is created and an enterprise application to which this Web application belongs is also created. If a Web application does not exist, then create it before you proceed with the instructions. If you have the WebSphere Commerce development environment setup, then do not create a new enterprise application and Web application. Instead, use the WebSphereCommerceServerExtensionsLogic project.

The following is an overview of the procedure to generate the client code and data types using IBM WebSphere Studio:

1. Create a new folder such as `wsdl` in the Web project where you want to import the WSDL definitions. If you do not have a Web project, then create a Web project.
2. Right click on the `wsdl` folder.
 - a. Select **Import** > **File System** and click **Next**.

- b. Click **Browse** and navigate to the folder where the WSDL definition files are stored.
- c. Click **OK**. You will find the WSDL definitions in the following directory:
 -  `WC_installdir\samples\webservices\xml\messaging`
 -  `WCStudio_installdir\Commerce\samples\webservices\xml\messaging`
 -    `WC_installdir/samples/webservices/xml/messaging`
 -  `/QIBM/ProdData/CommerceServer55/samples/webservices/xml/messaging`
3. From the right panel, select the WSDL files, which you want to import into WebSphere Studio and click **Finish**.

Generating the client code and data types

After you import the WSDL definitions into WebSphere Studio, you can generate the client proxy code and a sample test client to test the Web service. A client proxy is a Java class that can be used to access a Web service from a Java client. The sample test client provides a sample JSP, which enables testing the Web service from WebSphere Studio. Do the following to generate the client proxy and the sample test client:

Note: You can use the following set of instructions to generate the client code to access a Web service hosted by an external client from WebSphere Commerce. The only difference is that you must begin by obtaining the external service implementation definition from the service provider and import that WSDL definition into WebSphere Studio.

1. Move to Web perspective and select the Web project in which you want to save the generated client code. If you don't have any Web projects, then create a Web project.
2. Select and right click on the WSDL definition file to generate the client code. From the menu select **New > Other**. The New wizard displays.
- + 3. Select **Web services** from the left panel and **Web service client** from the right panel. Click **Next**.
- + 4. From the Web service client dialog select **Java proxy** and **Test the generated proxy**. Click **Next**.
- + 5. In the Web Service WSDL File Selection dialog ensure that the chosen WSDL file is selected and click **Next**.
6. Note the class name in the Web Service Binding Proxy Generation dialog and click **Next**.
- + 7. Ensure that the value of the test facility is set to Web service sample JSP files. Click **Finish**.
- + 8. Switch to the Web perspective and expand the Samples folder. You will see the test JSP clients.

Testing the Web service

This section explains how to test the Web services provided by WebSphere Commerce, using the sample test client that you generated in the preceding section. Do the following to test the Web service:

1. Ensure that at least one WebSphere Test Environment server exists. If you don't have a server, then create the server by doing the following:
 - a. Move to the server perspective.
 - b. Select **File > New > Server and Server Configuration**.
 - c. Enter the name of the server and select the type as **WebSphere version 5.0 > Test Environment**.
2. Right click TestClient.jsp and select **Run on Server**. The test client appears in a browser window.
3. Select the **getEndPoint()** method from the left frame and click **Invoke**. In the results frame, ensure that the end point returned is the URL of the server to which you want to connect.
4. If the end point returned does not match the server where the WebSphere Commerce server is running, then set the URL to point to the WebSphere Commerce Server:
 - a. Select the **setEndPoint()** method from the left frame.
 - b. Update the WebSphere Commerce server URL in the right frame and click **Invoke**.
5. Select the business method of the client proxy. For the OrderCreate proxy the business method is, orderCreate.
 - a. Type the details of the parameters to be passed in the form, which displays in the right frame.
 - b. Type the values for all mandatory fields.
 - c. Click **Invoke**. A SOAP message is composed and sent to the server identified by the endpoint.

You can see the results of the Web service invocation in the Result frame.

+
+
+
+
+

Note: When using WebSphere Studio to create clients for Web services, if the method signature contains an array, then the JSP client generated may be incomplete. Consequently, the client cannot invoke the Web service. For more information, refer to the WebSphere Studio documentation on the "Limitations of Web services".

+
+
+

The alternative method to test the Web services is to add a main method into the generated proxy code. This makes the proxy code a stand alone application, which can invoke the Web service.

Using the TCP Monitor to trace SOAP messages

This section describes how to create and configure a TCP/IP monitor server instance and how to use the TCP/IP Monitor server to trace SOAP messages. This helps when debugging a Web service or when defining a mapping template for a Web service that you are developing.

The TCP/IP Monitor is a simple server that monitors all the requests and responses between a Web browser and an application server. It can capture and display all traffic between the client and the application server.

Here, you need to create a new TCP/IP Monitor server instance and do the following:

- Configure the TCP/IP Monitor server to forward requests to the WebSphere Commerce server.

- Configure your test client to point to the TCP/IP Monitor so that you can intercept the SOAP requests and responses.

Creating a TCP Monitor server instance

To create a TCP/IP Monitor server instance called TCPMon, do the following:

1. From the main menu select **File > New > Other**.
2. Select **Server** from the left panel, **Server and Server Configuration** from the right panel and click **Next**.
3. In the Create a New Server and Server Configuration window do the following:
 - a. Type a **Server name**, for example, TCPMon.
 - b. Select the **Server type**, TCP/IP monitoring server.
 - c. Click **Next**.
4. In the Create a New Server and Server Configuration window do the following:
 - a. Type the WebSphere Commerce server host name in the **Remote host** field.
 - b. Type the port number at which the WebSphere Commerce server is listening in the **Remote port** field. You can use localhost as the hostname if WebSphere Commerce is running on the same machine as the client.
 - c. Click **Finish**.

Configuring the TCP Monitor server instance

Configure the TCP Monitor such that the WebSphere Commerce Server's listener port is the TCP Monitor's remote port. Do the following:

1. Move to the server perspective.
2. Double click on the TCP Monitor instance in the server configuration view. The TCP Monitor's server configuration opens in the editor view, as shown in the following figure:

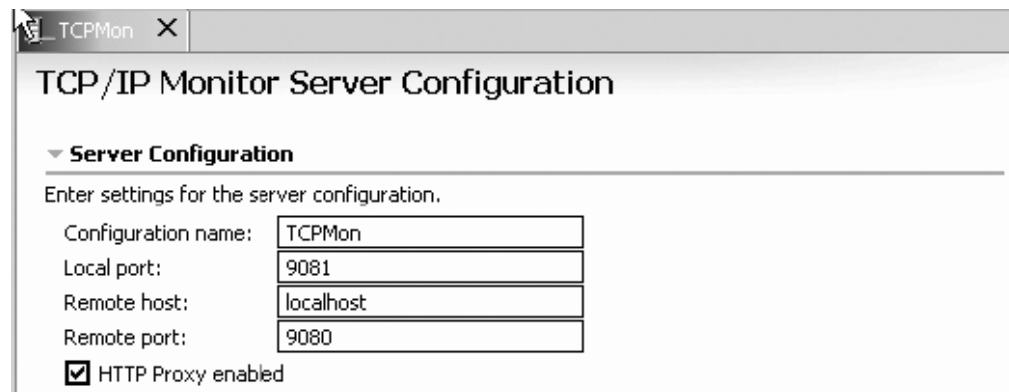


Figure 6. Configuring the TCP/IP Monitor Server Configuration

Here, the local port is the port at which the TCP monitor is listening and the remote port is the port at which the WebSphere Commerce server is listening.

Tracing the SOAP messages using the TCP/IP Monitor

After configuring the TCP/IP Monitor, use it to trace SOAP messages. To do this, start the sample test client generated on the server and configure it to send the request to the TCP/IP Monitor. Do the following:

1. Ensure that the TCP/IP Monitor and the WebSphere Commerce server are running.

- a. If the WebSphere Commerce server is running outside WebSphere Studio, then start the server using the WebSphere Application Server administration console.
 - b. If the WebSphere Commerce server is running inside WebSphere Commerce Studio, then move to the **Server** perspective. Open the **Servers** view. If the **Status** of the WebSphereCommerceServer server is Stopped, then right click on the server and select **Start**. Wait until the Status changes to Started in the Servers view.
2. Move to the Web perspective. **Browse** to sample\TestClient.jsp. Right-click sample\TestClient.jsp and select **Run on server**.
 3. In the server selection window, ensure that the server on which the JSP file has to run is selected. Click **Finish**.
 4. If you are prompted to select a server client, then select the **Web Browser** and click **Finish**. A new browser window opens with three frames as shown in the following figure:

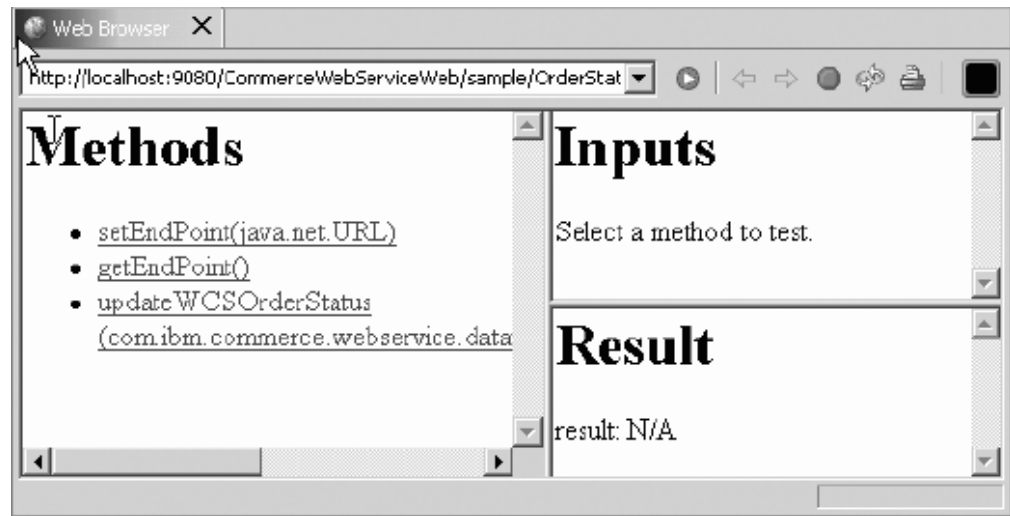


Figure 7. Selecting a Web browser

- a. Select the **getEndPoint()** method from the Methods frame as shown in the following figure:

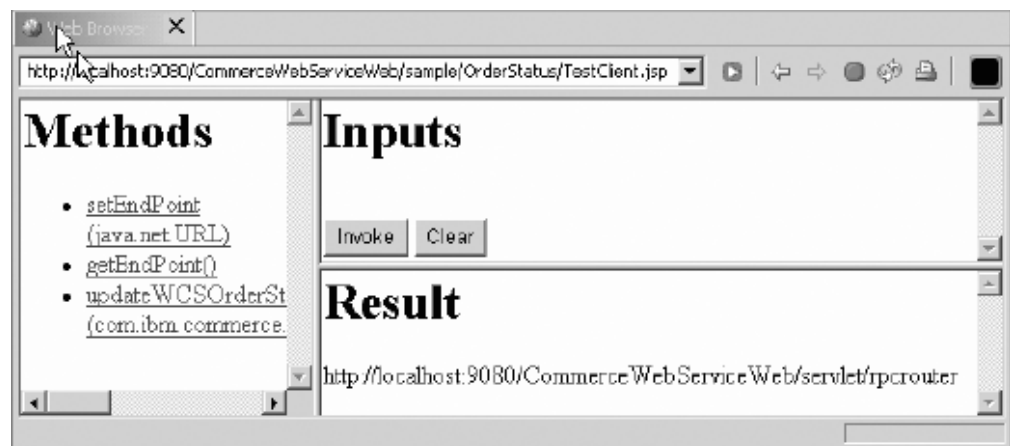


Figure 8. Selecting the getEndPoint() method

- b. Click **Invoke** in the Inputs frame. The results display in the Results frame
5. Configuring the test client to send the request to the TCP Monitor involves the following:
 - a. Select the **setEndPoint()** method from the Methods frame.
 - b. In the Inputs frame enter the **url** of the TCP Monitor. For example, `http://localhost:9080/webapp/wcs/stores/servlet/`, where 9080 is the port at which the TCP Monitor is running and `webapp/wcs/stores/servlet/` is the Web path where the WebSphere Commerce request servlet.

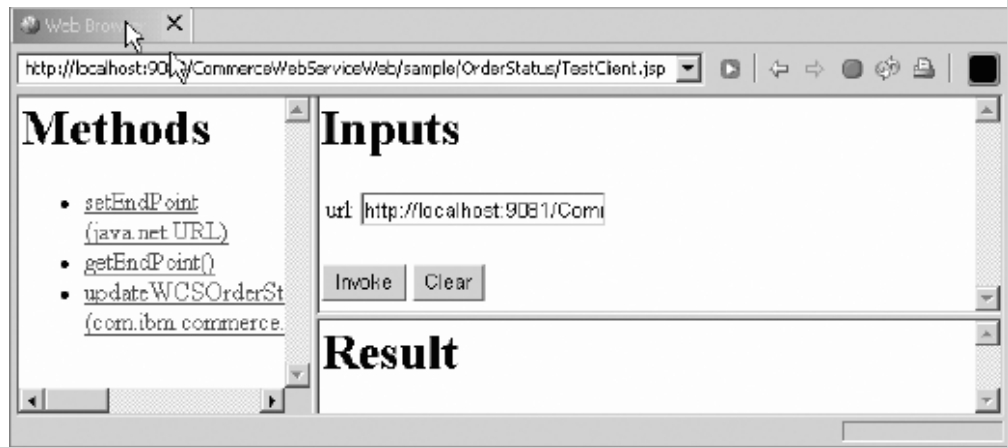


Figure 9. Selecting the `setEndPoint()` method

- c. Click **Invoke**.
6. Invoke the **getEndPoint()** method again to verify the endpoint.
7. To invoke the business logic do the following:
 - a. Click the business logic method, for example, **updateWCSOrderStatus()** in the Methods frame.
 - b. Enter the values for all mandatory parameters in the Inputs frame as shown in the following figure:

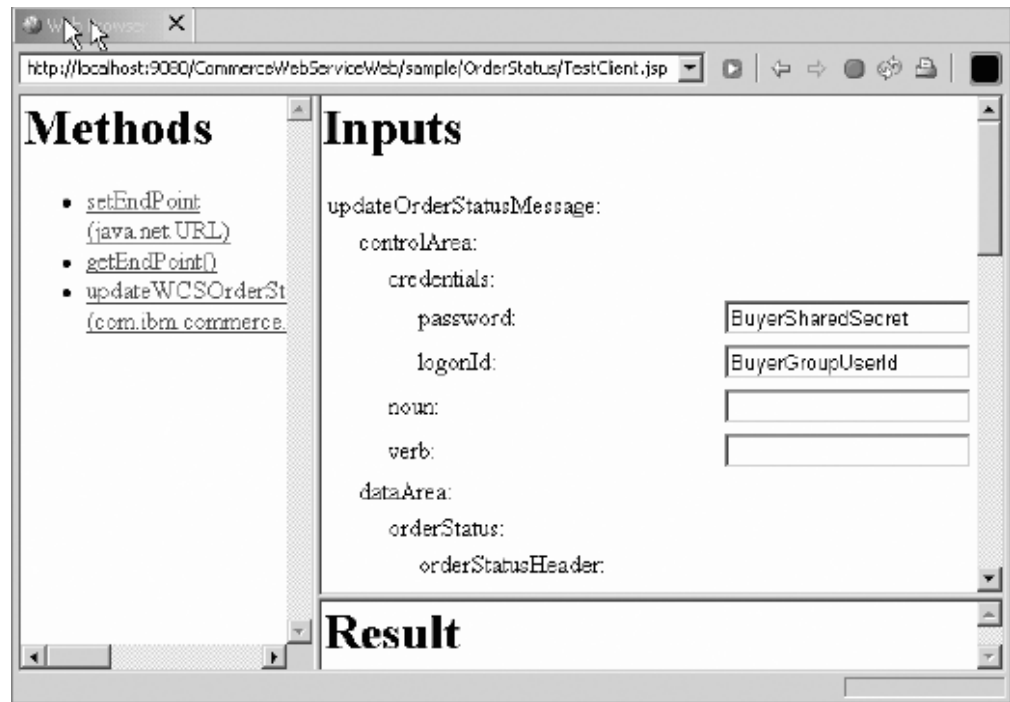


Figure 10. Selecting the business logic

- c. Click **Invoke**.
- d. The results display in the results frame.
- e. Move to the TCP/IP Monitor view and select the latest message. The SOAP Request and Response now displays as shown in the following figure:

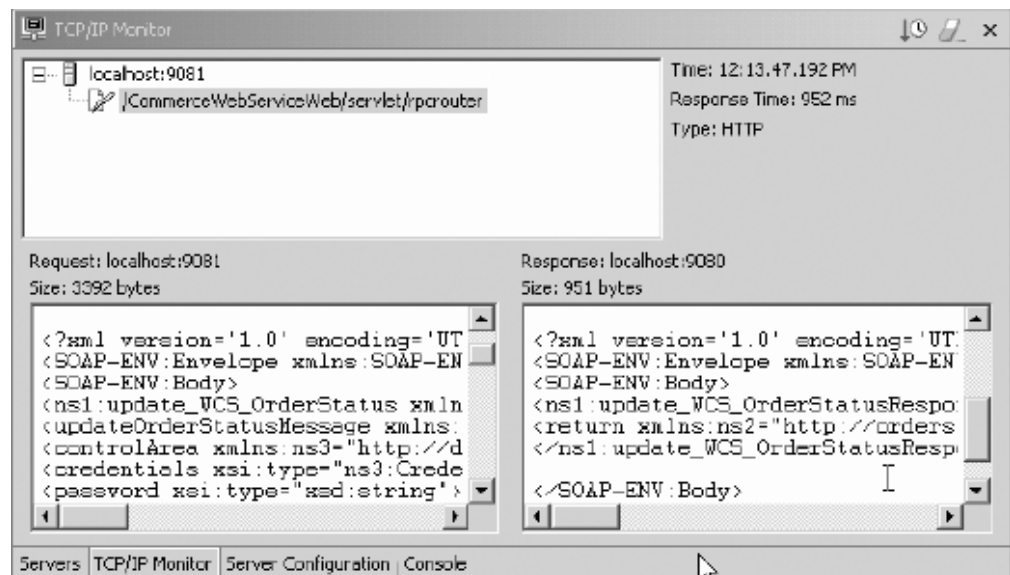


Figure 11. Displaying SOAP Request and Response

Customizing the client code

When generating the client code, you must consider that the default client code generated may not always work with WebSphere Commerce. For example, if you are using the default `BeanSerializer` class to serialize the data types, some of the data types will be converted to XML strings that WebSphere Commerce may not understand at the server end. For example, the `OrderCreate` sample Web service contains such a data structure. This data structure is provided as an interface to the WebSphere Commerce command and is called `UserData` data structure. The `UserData` data structure allows customized data to be passed to the command using the message-driven interface.

The following section describes how to serialize the `UserData` data structure so that WebSphere Commerce can use it.

Note: The `UserData` data structure provided is a sample and is optional for the `OrderCreate` Web service.

WebSphere Commerce allows external systems to include custom name-value pairs as a part of the XML message that is sent. To support this, an XPATH type called `USERDATA` is used, which will be processed as described by the following sample XML message that includes the `UserData` element:

```
<UserData>
  <UserDataField name="abc">xyz</UserDataField>
</UserData>
```

The XPATH definition for this `UserData` element is as follows:

```
<Tag XPath='.../UserData/UserDataField'
  XPathType='USERDATA' />
```

The XPATH processor translates the `UserData` element as the name-value pair `"abc=xyz"`.

If you have a data structure similar to the following, then there could be a problem with the default bean serializer that comes with the Apache SOAP implementation. The Apache SOAP implementation converts all the datatypes in a data structure to separate elements in the XML document. If you are using the Apache SOAP client to convert your Java objects into a SOAP XML message, then WebSphere Commerce will be unable to parse this XML and therefore, unable to form a `UserData` object.

```
class UserData
(
  private String userDataField = null;
  private String name = null;
  // getter and setter methods here ...
)
```

This class will be serialized into an XML structure.

```
<UserData>
  <UserDataField>xyz</UserDataField>
  <name>abc</name>
</UserData>
```

As a result, the WebSphere Commerce message mapper will not be able to understand this as a `USERDATA` structure.

In order to support the serialization of the `UserData` datatype into the required format, you must write a custom serializer and register it with the SOAP mapping registry. The custom serializer must have the following `marshall` method:

```

/*
 * serialize the com.ibm.commerce.webservices.datatype.UserData
 * to the corresponding XML schema as expected by WC.
 *
 * @param String inScopeEncStyle
 * @param Class javaType
 * @param Object src
 * @param Object context
 * @param Writer sink
 * @param NSStack nsStack
 * @param XMLJavaMappingRegistry xjmr
 * @param SOAPContext ctx
 * @exception IllegalArgumentException
 * @exception IOException
 */
public void marshall(String inScopeEncStyle,
                    Class javaType,
                    Object src,
                    Object context,
                    Writer sink,
                    NSStack nsStack,
                    XMLJavaMappingRegistry xjmr,
                    SOAPContext ctx)
    throws IllegalArgumentException, IOException
{
    if (!javaType.equals(UserData.class))
    {
        throw new IllegalArgumentException(
            "Can only serialize UserData instances");
    }
    nsStack.pushScope();
    if (src != null)
    {
        SoapEncUtils.generateStructureHeader(inScopeEncStyle,
                                           javaType,
                                           context,
                                           sink,
                                           nsStack,
                                           xjmr);

        UserData data = (UserData)src;
        sink.write("<UserDataField name=\"" + data.getName() + "\">");
        sink.write(data.getValue() +
            "</UserDataField>");
        sink.write("</" + context + ">");
    }
    else
    {
        SoapEncUtils.generateNullStructure(inScopeEncStyle,
                                           javaType,
                                           context,
                                           sink,
                                           nsStack, xjmr);
    }
    nsStack.popScope();
}

```

Note: The sample service provided with WebSphere Commerce Version 5.5 is designed to work with the preceding code.

Notices

This information was developed for products and services offered in the U.S.A. IBM may not offer the products, services, or features discussed in this document in other countries. Consult your local IBM representative for information on the products and services currently available in your area. Any reference to an IBM product, program, or service is not intended to state or imply that only that IBM product, program, or service may be used. Any functionally equivalent product, program, or service that does not infringe any IBM intellectual property right may be used instead. However, it is the user's responsibility to evaluate and verify the operation of any non-IBM product, program, or service.

IBM may have patents or pending patent applications covering subject matter described in this document. The furnishing of this document does not give you any license to these patents. You can send license inquiries, in writing, to:

IBM Director of Licensing
IBM Corporation
North Castle Drive
Armonk, NY 10504-1785
U.S.A.

For license inquiries regarding double-byte (DBCS) information, contact the IBM Intellectual Property Department in your country or send inquiries, in writing, to:

IBM World Trade Asia Corporation
Licensing
2-31 Roppongi 3-chome, Minato-ku
Tokyo 106-0032, Japan

The following paragraph does not apply to the United Kingdom or any other country where such provisions are inconsistent with local law:

INTERNATIONAL BUSINESS MACHINES CORPORATION PROVIDES THIS PUBLICATION "AS IS" WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESS OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF NON-INFRINGEMENT, MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE. Some states do not allow disclaimer of express or implied warranties in certain transactions, therefore, this statement may not apply to you.

This information could include technical inaccuracies or typographical errors. Changes are periodically made to the information herein; these changes will be incorporated in new editions of the publication. IBM may make improvements and/or changes in the product(s) and/or the program(s) described in this publication at any time without notice.

IBM may use or distribute any of the information you supply in any way it believes appropriate without incurring any obligation to you.

Licensees of this program who wish to have information about it for the purpose of enabling: (i) the exchange of information between independently created programs and other programs (including this one) and (ii) the mutual use of the information that has been exchanged, should contact:

IBM Canada Ltd.
Office of the Lab Director
8200 Warden Avenue,
Markham, Ontario L6G 1C7
Canada

Such information may be available, subject to appropriate terms and conditions, including in some cases, payment of a fee.

The licensed program described in this document and all licensed material available for it are provided by IBM under terms of the IBM Customer Agreement, IBM International Program License Agreement or any equivalent agreement between us.

This document may contain information about other companies' products, including references to such companies' Internet sites. IBM has no responsibility for the accuracy, completeness, or use of such information.

COPYRIGHT LICENSE:

This information contains sample application programs in source language, which illustrates programming techniques on various operating platforms. You may copy, modify, and distribute these sample programs in any form without payment to IBM, for the purposes of developing, using, marketing or distributing application programs conforming to the application programming interface for the operating platform for which the sample programs are written. These examples have not been thoroughly tested under all conditions. IBM, therefore, cannot guarantee or imply reliability, serviceability, or function of these programs. You may copy, modify, and distribute these sample programs in any form without payment to IBM for the purposes of developing, using, marketing, or distributing application programs conforming to IBM's application programming interfaces.

Trademarks

IBM, and the IBM logo are trademarks or registered trademarks of International Business Machines Corporation in the United States and/or foreign countries.

The following terms are trademarks or registered trademarks of International Business Machines Corporation in the United States or other countries or both:

AS/400	iSeries
AIX	@server
DB2	S/390
DB2 Universal Database	zSeries
IBM	WebSphere

Microsoft[®], Windows, and Windows NT[®] are trademarks or registered trademarks of Microsoft Corporation.

Java, and all Java-based trademarks are trademarks of Sun Microsystems, Inc. in the United States, other countries, or both.

Other company, product, or service names may be trademarks or service marks of others.



Printed in USA