IBM WebSphere Commerce

Connectivity and Notification: Online Help Files

Version 5.4

IBM WebSphere Commerce

Connectivity and Notification: Online Help Files

Version 5.4

Note!

Before using this information and the product it supports, be sure to read the general information in the Notices section.

Contents

Chapter 1. Program Adapter

The Program Adapter 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 with a common way to communicate with WebSphere Commerce through HTTP, allowing WebSphere Commerce to act as a supplier to these systems, for buyer/supplier transactions. The Program Adapter handles incoming XML requests by performing the following actions:

- Recognizing the request and verifying if it is an XML request. If the following three attributes of the request are supported, it can be distinguished it as an XML request.
  - content-type
  - method
  - character encoding

  The supported request attributes are specified in the adapter configuration.
- Extracting the input stream of the request.
- Calling the message mapper and passing the content of the input stream.
- Receiving the CommandProperty object representing a WebSphere Commerce command returned by the message mapper.
- Determining the proper device format in which to generate the response.
- Executing the command.
- Sending an XML response message, created by a JSP and based on the viewname specified by the command executed and the device format of the received request.

Each request is treated as a separate session. The credentials of the message are specified in the control area of the message. By default, the Program Adapter checks the user ID and password parameters to determine the authenticity of a request. The Program Adapter does not support legacy messages because legacy messages do not support the specification of credentials.

The lifecycle of the Program Adapter exists throughout the WebSphere Commerce instance. It is initialized when an instance is started unless its configuration parameters are removed or the adapter is not enabled, and it resides as long as the instance runs.

For architectural information on how WebSphere Commerce handles receiving requests from devices, refer to the *WebSphere Commerce Programmer's Guide*.

CommandProperty object

The CommandProperty object is a representative of a controller command. The object contains the command name to be executed, the command properties when executing the command, and the parameters of the command. The purpose of the message mappers is to convert inbound request messages into controller commands to be executed by an adapter. Although they can be used by all components of WebSphere Commerce to map data into a extended TypedProperty object, the main purpose for message mappers is converting XML objects into common Java objects that represents controller commands.

The CommandProperty datatype is composed of the following three parts:

- `commandName`: the name of the command to be executed, in the form of a string.
- `requestProperties`: the command properties when executing the command, in the form of a TypedProperty object.
- `executionProperties`: the control data for executing the command in the form of a TypedProperty object.

Device format algorithm

The device format algorithm is used to determine the appropriate JSP to use as the response for a particular request. When a request is received and the message mapper is used to convert the request into a CommandProperty object, the message mapper and adapter used to process the request determine the appropriate device format for generating the content of the response. The ID of the message mapper is added to the device format ID of the adapter to determine the overall device format ID of the response. This overall device format ID and the VIEWNAME are used to get the appropriate JSP from the VIEWREG table, which generates the content of the response.

Each adapter accepting requests and using the message mapper is given a device format ID. This ID is defined in the adapter's *instance_name*.xml configuration file. This ID is defined in the adapter's configuration found in the *instance_name*.xml configuration file. The default device format ID for each receiving adapter using the message mapper is in intervals of -10000. An program adapter has a device format ID of -10000 and the MQSeries adapter has a device format ID of -20000. The adapter that supports legacy messages uses the device format ID of -30000. When determining the appropriate JSP and view command to call the JSP, the message mapper ID is added to the adapter's device format ID to determine the device format ID for the response view. However, if the view of the calculated device format ID does not exist, the default view of the adapter's interval is used. Thus, for an XML over HTTP request, the default device format ID is -10000 and for MQSeries adapter requests, the default device format ID is -20000. Differentiating the response view is necessary so that an HTTP response will use the `HttpForwardViewCommand` interface while a response from the MQSeries adapter will use the `MessagingViewCommand` interface.

This algorithm allows a maximum of 9999 possible message mappers to be defined with one adapter. The configuration file for each message mapper contains an ID number in intervals of 1. To configure additional adapters that use the message mapper should be given a device format ID that is an interval of 10000. For example, a new request mechanism could be given a device format interval of 40000.

Note 1: If the calculated device format ID cannot be found, the default device format ID of the adapter is used to obtain the result. This number is configurable in the HTTP adapter configuration.

Note 2: The size of the interval is not important. For example, if only 2 message mappers exist, then the interval can be of size 3. The interval 10000 is used by default.

Example 1: An inbound message is handled by a message mapper with an ID of -1 and by the program adapter which has a device format ID of -10000. Following the device format algorithm, the device format ID generated for a response to that inbound message would be -10001. The JSP file defined in the VIEWREG table for the VIEWNAME and the device format ID of -10001 is used for creating the response.

Example 2: Alternatively, an inbound message is handled by the same message mapper with the ID of -1, but is handled by the MQSeries adapter, which has a device format ID of -20000. The generated device format ID for the response to that message would be -20001. Thus, the JSP file defined in the VIEWREG table for the VIEWNAME and the device format ID of -20001 is used for creating the response. Note that even though the request can use the same JSP for responding to the request, the class used to call the JSP may differ.

XML over HTTP

WebSphere Commerce can receive inbound XML messages over HTTP using the Program Adapter. The following steps illustrate the overall flow of an XML over HTTP request:

1. An external system sends an XML message to WebSphere Commerce over HTTP.
2. The request is mapped to the Program Adapter.
3. The Program Adapter passes the XML request to the appropriate message mapper.

4. The message mapper converts the XML request into a CommandProperty object and passes it back to the Program Adapter.
5. The Program Adapter prepares the command for execution and passes it to the WebController for execution.
6. The Program Adapter generates the proper XML response and returns the XML response to the external system that made the request.

When the Program Adapter receives the XML request, it must verify the credentials of the external system that sent the request. Not all XML requests can be processed. Even if the XML request can be mapped to a WebSphere commerce command, there must be some verification to ensure that the request should be processed. See Program Adapter Security for HTTP Requests for more information.

MQSeries as middleware

The MQSeries adapter allows you to integrate back-end and external systems with WebSphere Commerce using MQSeries as middleware. The MQSeries adapter allows WebSphere Commerce to receive messages from back-end systems and external systems. The supported software is MQSeries Version 5.2 or higher, with the MA88 product extension.

You can set up MQSeries as your middleware through the use of MQ Java in one of two modes:

- **bindings mode**
  WebSphere Commerce is installed on the same machine as the MQSeries server and it connects to the MQSeries server through MQSeries Java using the Java Native Interface (JNI). Since communication is through direct JNI calls to the queue manager API rather than through a network, bindings mode provides better performance than client mode done using network connections.
- **client mode**
  WebSphere Commerce is installed on one machine and the MQSeries server is installed on a back-end system.

To verify MQSeries connections, queues, and channels, run test programs to put messages into and get messages from MQSeries queues. For details, refer to your MQSeries documentation.

WebSphere CommerceMQSeries adapter

The WebSphere Commerce MQSeries adapter, or simply the MQSeries adapter, is a component of WebSphere Commerce that enables integration with back-end systems by processing inbound messages via MQSeries. The MQSeries adapter is a combination of the JMS-MQ CCF Connector to retrieve MQ messages, and the Program Adapter which is called to execute those messages.

The MQSeries adapter has a set of predefined messages that help integrate WebSphere Commerce business processing with back-end or external system processing. Each incoming message activates processes within WebSphere Commerce to update database tables or perform other operations. Refer to the back-end integration and fulfillment integration message information for more details on the messages provided. In addition to the existing pre-defined messages, the adapter supports message extensions and new messages.

Parallel versus serial message processing in MQSeries adapter

MQSeries adapter can process inbound messages in two ways: serially or in parallel. Serial processing means that each message is put in a line, or queue, and handled one after another. In this method, each message must wait until processing of the previous message is complete. Parallel processing on the other hand, means that a number of messages can be processed at the same time. Instead of each message having to wait for the previous one to be completed, many of them can be run simultaneously.

Although parallel processing generally results in faster throughput, it is not suitable for all types of requests. There are some situations in which the serial nature of the transactions must be preserved. For example, if a new customer registers at your store, then makes a correction to their address information, then makes a purchase order, you would want the order of these transactions to be preserved when processing them. You could not perform the address modification or the purchase order unless the account had already been created. Likewise, you would not want to fulfill a purchase order without having the correct shipping information.

Although it is generally preferable to use parallel processing where possible, you will have to decide where it is appropriate to use this method on your data.

Chapter 2. Configuring the Program Adapter

The program adapter is configured using the *instance_name*.xml configuration file. The program adapter entry is used to define the adapter. The information found between the ProgramAdapter nodes defines the configuration of the program adapter, defining which message mappers it uses, the supported content-types, and other request attributes that distinguish the request as an XML over HTTP request. In the *instance_name*.xml configuration, the entry for the program adapter should look similar to the following:

```
<HttpAdapters display="false">
    <HttpAdapter deviceFormatTypeId="-10000"
        enabled="true"
        deviceFormatId="-10000"
        deviceFormatType="XmlHttp"
        factoryClassname="com.ibm.commerce.programadapter.HttpProgramAdapterImpl"
        name="XML/HTTP">
    <ProgramAdapter>
    <SessionContext
class="com.ibm.commerce.messaging.programadapter.security.CredentialsSpecifiedProgramAdapterSessionContextImp
    <SessionContextConfig />
    </SessionContext>
    <Configuration supportedMethods="POST, M-POST"
        supportedContentTypes="text/xml, text/xml-SOAP"
        supportedMessageMappers="WCS.INTEGRATION"
        supportedCharacterEncoding="ISO8859-1, UTF-8" />
    </ProgramAdapter>
    </HttpAdapter>
</HttpAdapters>
```

Enabling the Program Adapter for XML Requests over HTTP

When the instance is created, the Program Adapter is disabled. In order to support XML over HTTP, you must enable the Program Adapter using the *instance_name*.xmlconfiguration file. To enable the Program Adapter to support XML requests over HTTP, do the following:

1. Open the *instance_name*.xml configuration file.
2. Locate the HttpAdapterssection and set the enabled parameter to "true".

Adding adapters

To add a new adapter, you must manually add it to the group of adapters in the *instance_name*.xml configuration file. To add a new adapter, do the following:

1. Open the *instance_name*.xml configuration file.
2. Locate the HttpAdapters XML node and add XML syntax similar to the following to define your adapter:

```
<HttpAdapter
enabled="true/false"
deviceFormatType="–device format–"
deviceFormatId="#"
name="–name–"
factoryClassname="– class implementing HttpAdapterFactory –">
< –– free range format of XML to contain adapter configuration information –>
```

Note: The above syntax would be used to add a program adapter. To add a different type of adapter, modify the class implementation accordingly. In the above example, `HttpAdapterFactory` is the class implementation.

Downloading and installing the MQSeries MA88 product extension pack

To install the MQSeries MA88 product extension pack, do the following:

Note: This information is subject to change as the MQSeries installation procedure is updated.

Windows

AIX

Solaris

1. Download the appropriate ma88 product extension pack for your operating system from the following URL:
   `http://www.ibm.com/software/ts/mqseries/txppacs/ma88.html`. The file is in compressed (zipped) format.
   - Download the document *MQSeries Using Java* in PDF format from the same URL.
2. Decompress and install SupportPac.
3. Follow the setup instructions, installing the product extensions in the *`MQ_install_path`*`\java` directory, where *`MQ_install_path`* is the path where MQSeries is installed.
4. Update the `admin.config` file found in the following directory:

   2000

   `drive``:\Program Files\WebSphere\CommerceServer\bin`

   NT

   `drive``:\WebSphere\CommerceServer\bin`

   AIX

   `/usr/WebSphere/CommerceServer/bin`

   Solaris

   `/opt/WebSphere/CommerceServer/bin`

   400

   `/QIBM/Proddata/WebCommerce/bin`
5. Add the *`MQ_install_path`*`\java\lib` directory to the `com.ibm.ejs.sm.util.process.Nanny.path` variable. See Updating the WebSphere Application Server classpath variable for changes to iSeries class libraries.

400

1. Download the appropriate ma88 product extension pack for your operating system from the following URL:
   http://www.ibm.com/software/ts/mqseries/txppacs/ma88.html. The file is in compressed (zipped) format. Be sure to get the file named `ma88_iSeries.zip`.
   - Download the document MQSeries Using Java in PDF format from the same URL.

2. Uncompress using InfoZip's Unzip. This will create the file `ma88_400.sav`.
3. Create a save file called MA88 in a suitable library on the iSeries 400, for example QGPL
   `CRTSAVF FILE(QGPL/MA88)`
4. Transfer `ma88_iSeries.sav` into this save file as a binary image. If you use FTP to do this, the put command should be similar to:
   `PUT C:\TEMP\MA88_iSeries.SAV QGPL/MA88`
5. Install the MQSeries classes for Java, product Id 5648C60, using RSTLICPGM:
   `RSTLICPGM LICPGM(5648C60) DEV(*SAVF) SAVF(QGPL/MA88)`
6. Delete the save file created in Step 2:
   `DLTF FILE(QGPL/MA88)`

Enabling the MQSeries adapter

Use the following checklist to enable the MQSeries adapter messages.

1. Install MQSeries Version 5.2. Refer to the document *MQSeries Using Java* for information on how to set up either the MQSeries binding mode or the MQSeries client mode configuration. For e-Integrator Version 3.0 use the bindings mode.
   - Windows

     AIX

     Solaris

     The user logon ID must have the authority to read and write to the queue manager and queues defined. For e-Integrator Version 3.0 use the binding mode.
   - 400

     The Instance User Profile must have the authority to read and write to the queue manager and queues defined. To define this authority use the `GRTMQMAUT` command.
2. Ensure that the following MQSeries objects have been defined:
   - Queue manager
   - Inbound message queue
   - Outbound message queue
   - Transmission queue
   - Error queue
   - Parallel queue
   - Serial queue
3. Windows

   AIX

   Solaris

   Set the MQSeries queue manager coded character set identifier to 1208 (UTF8). Run the following MQSeries commands from the command line:

```
strmqm YourQueueManagerName
runmqsc YourQueueManagerName
alter qmgr ccsid(1208)
end
```

where *YourQueueManagerName*is the name of your MQSeries queue manager.

4. If you are using the MQSeries client mode, all of the required channels must be defined and you must identify the channel name that the MQSeries client will use to communicate with the MQSeries server.
5. Download and install the MQSeries MA88 product extension pack.
   This product extension contains the Java Message Service (JMS) API that the MQSeries adapter uses to communicate with MQSeries.
6. Configure JMS for MQSeries.
   You must create a JMS QueueConnectionFactory and JMS Queues that map to corresponding MQSeries objects. This allows the MQSeries adapter to access MQSeries entities through JMS.
7. Configure JMS using JMSAdmin.
8. Enable the messaging system transport adapter.
9. Update the WebSphere Application Server classpath variable.

Note: To use the MQSeries adapter, ensure that the Queue Manager is started before you start the WebSphere Commerce Server and instance.

Configuring JMS for MQSeries

To configure the messaging system to work with JMS (Java Messaging Service), do the following:

1. 400

   The iSeries QShell provides an emulator where Unix commands can be executed on your WebSphere Commerce machine.You must use Java 1.3 to perform the remaining commands. To set the user profile to use Java version 1.3, add the line `java.version=1.3` into the users `SystemDefault.properties` file. For more information, consult the iSeries Java documentation. You must do this before starting QShell.

2. 400

   Start the iSeries QShell by typing: `STRQSH` at a CL prompt.

3. Windows

   AIX

   Solaris

   From the WebSphere Commerce machine update your `classpath` variable:

   a. Type the following command, all on one line:

      Windows

      ```
      set classpath=%classpath%;MQ_install_path\java\lib\com.ibm.mqjms.jar;
      MQ_install_path\java\lib\com.ibm.mq.jar;WAS_install_path\lib\ns.jar
      ```

      AIX

Solaris

```
export CLASSPATH=$CLASSPATH:MQ_install_path/java/lib/com.ibm.mqjms.jar:
MQ_install_path/java/lib/com.ibm.mq.jar:WAS_install_path/lib/ns.jar

export
CLASSPATH=$CLASSPATH:WAS_install_path/lib/ujc.jar:
WAS_install_path/lib/ejs.jar:
WAS_install_path/lib/sslight.jar
```

Note: This statement is too long to add as one statement at the command prompt. It must be added in two segments.

where

WAS_install_path is the path in which you installed the WebSphere Application Server

where *MQ_install_path* is the path in which you installed MQSeries.

b. Add a new environment variable named `MQ_JAVA_INSTALL_PATH` by typing the following command:

Windows

```
set MQ_JAVA_INSTALL_PATH=MQ_install_path\java
```

AIX

Solaris

```
export MQ_JAVA_INSTALL_PATH=MQ_install_path/java
```

where *MQ_install_path* is the path in which you installed MQSeries.

Update the environment to use the jdk that comes with WebSphere Application Server by typing the following command:

Windows

```
set PATH=WAS_install_path\Java\bin;%PATH%
```

AIX

```
export PATH=WAS_install_path/java/jre/sh:$PATH
```

Solaris

```
export PATH=WAS_install_path/java/jre/bin:$PATH
```

4. Configure JMS using JMSAdmin.

Updating the WebSphere Application Server classpath variable

To update the WebSphere Application Server `classpath` variable for an instance, do the following:

1. Open the WebSphere Application Server Advanced Administrative Console.
2. Select the host on which you are running your WebSphere Commerce instance.
3. Select **WebSphere Administrative Domain**.
4. Select **Nodes**.
5. Select your *host name*.
6. Select **Application Servers**

7. Select WebSphere Commerce Server *instance_name*,
   where *instance_name* is the name of your WebSphere Commerce instance. For iSeries select *instance_name* - WebSphere Commerce Server.
8. Go to the JVM Settings table of the instance.
9. Select **Add** to add a new system property.
10. Type in the following system property:
    `name = ws.ext.dirs`
    `value = MQJAVA/lib` For iSeries: `value=/QIBM/ProdData/mqm/java/lib`
11. Click **Apply**to apply the changes.
12. Repeat steps 2-7 for every WebSphere Commerce instance with which MQSeries is used.
13. Close the WebSphere Advanced Administrative Console.

400

For iSeries perform steps 1-11 above, then do the following:

1. Click **JVM.**
2. To the right of the System Properties box, click **Add**. A new system property shows up in the list.
3. Type `java.library.path` under the **Name** field.
4. Type `/QSYS.LIB/QMQMJAVA.LIB` under the **Value** field.
5. Click **Apply**to apply the changes.
6. Repeat steps 2-7 for every WebSphere Commerce instance with which MQSeries is used.
7. Close the WebSphere Advanced Administrative Console.

Configuring JMS using JMSAdmin

To map the queue manager and the queues created in the WebSphere Commerce namespace, do the following in QShell:

1. Ensure that WebSphere Application Server is running and that the environment variables and classpath have been set.
2. Change to the following directory:

   Windows

   MQ_install_path\java\bin

   AIX

   MQ_install_path/java/bin

   Solaris

   MQ_install_path/java/bin

   400

   /QIBM/ProdData/WebCommerce/bin

   where *MQ_install_path* is the path where MQSeries is installed.
3. Open the `JMSAdmin.config` file in a text editor.
   Ensure that the following three variables have been set to the values shown:
   `INITIAL_CONTEXT_FACTORY=com.ibm.ejs.ns.jndi.CNInitialContextFactory`
   `PROVIDER_URL=iiop://host_name:was_port`
   `SECURITY_AUTHENTICATION=none`
   where

host_name
Your instance host name

was_port
The WebSphere Application Server administration port used to configure the instance.

4. From a command line, run the JMSAdmin program:

   Windows

   ```
   JMSAdmin -cfg JMSAdmin.config -t -v
   ```

   AIX

   ```
   ./JMSAdmin -cfg JMSAdmin.config -t -v
   ```

   Solaris

   ```
   ./JMSAdmin -cfg JMSAdmin.config -v
   ```

   400

   ```
   ./JMSAdmin -cfg JMSAdmin.config -v
   ```

   Wait for the administration command line interface to load and the `Initctx>` prompt to appear.

5. Register the queue connection factory to the queue manager in the WebSphere Application Server namespace:

   `define qcf(`*JMSQueueConnectionFactory*`) qmanager(`*YourQueueManagerName*`)`

   where

   JMSQueueConnectionFactory
   This is defined in the QueueConnectionFactory ConnectionSpec attribute found in the JMS configuration for MQSeries. This can be found on the ConnectionSpec-JMS Interface CCF Connection page in the Configuration manager.

   YourQueueManagerName
   The name of your MQSeries queue manager.

6. Windows

   AIX

   Solaris

   Set the coded character set identifier to 1208 (UTF8):

   ```
   alter qcf(JMSQueueConnectionFactory) ccsid(1208)
   ```

   where

   JMSQueueConnectionFactory
   The name of the MQQueueConnectionFactory JMS object.

7. Define the following JMS queues. To define the queue type the appropriate command all on one line.
   JMSSerialInboundQueue — serial inbound queue

   ```
   define
   q(JMSSerialInboundQueue)qmanager(YourQueueManagerName)queue(YourSerialInboundQueueName)
   ```

   For example:

   ```
   define q(JMSSSerialInboundQueue)qmanager(WCSQMGR)queue(JMSSIBQ)
   ```

- **JMSParallelInboundQueue** — the parallel inbound queue
  `define q(JMSParallelInboundQueue)qmanager(YourQueueManagerName)queue(YourParallelInboundQueueName)`
- **JMSInboundQueue** — the inbound queue
  `define q(JMSInboundQueue)qmanager(YourQueueManagerName)queue(YourInboundQueueName)`
- **JMSOutboundQueue** — the outbound queue
  `define q(JMSOutboundQueue)qmanager(YourQueueManagerName)queue(YourOutboundQueueName)`
- **JMSErrorQueue** –the error queue
  `define q(JMSErrorQueue)qmanager(YourQueueManagerName)queue(YourErrorQueueName)`

  YourQueueManagerName
  The name of the MQSeries queue manager.

  YourSerialInboundQueueName
  The name of the MQSeries queue created for the serial inbound queue.

  YourParallelInboundQueueName
  The name of the MQSeries queue created for the parallel inbound queue.

  YourInboundQueueName
  The name of the MQSeries queue created for the inbound message queue.

  YourOutboundQueueName
  The name of the MQSeries queue created for the outbound queue.

  YourErrorQueueName
  The name of the MQSeries queue created for the error queue.

The default name of the queue is the same as the name used for the ConnectionSpec-JMS Interface CCF Connection values in Configuration Manager. If you change the default name, you must also change the name in Configuration Manager or for outbound messaging, the name change can be done in the Administration Console.

8. The outbound queue and error queue require that you set the target client to indicate that JMS is interacting with a native MQSeries application. Run the following command:
   `alter q(JMSOutboundQueue) targclient(MQ)`
   `alter q(JMSErrorQueue) targclient(MQ)`
9. Windows

   AIX

   Solaris

   400

   If you are using an MQSeries client/server setup where the MQSeries client is on the same machine as WebSphere Commerce Server, run the following commands:
   `alter qcf(`*`JMSQueueConnectionFactory`*`) transport(CLIENT)`
   `alter qcf(`*`JMSQueueConnectionFactory`*`) hostname(`*`YourMQServerHostName`*`)`
   where *YourMQServerHostName* is the name of your MQSeries server.
   This command connects to the remote MQSeries Server and configures the client.
10. Type `end` to exit the administration command line interface.

Response processing using MQSeries adapter

The MQSeries adapter can be used as an alternative interface to the WebSphere Commerce system and functionality. The MQSeries adapter not only processes inbound requests, but can provide a reply to the request. Any command that formulates a response can be configured to provide a response through the MQSeries adapter.

When an inbound message is received by the MQSeries adapter it is mapped to a command and that command is called with the parameters specified in the inbound message. If a response is to be generated from this inbound request, an entry in the VIEWREG table is needed. For more information on how the VIEWREG table is used, refer to the *WebSphere Commerce Programmer's Guide*. If a response is generated by the command, that response is placed on the outbound queue of the Parallel or Serial connector depending on the connector which receives the inbound request.

If an error with the inbound request occurs, the message generating the error is placed on the error queue. To generate an error response to the outbound queue, another entry is needed in the VIEWREG table to map the error response using the proper device format.

To setup the MQSeries adapter component for providing responses for inbound messages, do the following:

1. Verify that the *instance_name.xml* file includes an Outbound Queue for Inbound Parallel and Serial Connector by ensuring that syntax similar to the following is included for the Inbound Parallel Connector:

   ```
   <EditableProperty Admin="outQueue"
   editable="Yes"
   name="setOutboundQueue"
   display="false"
   value="JMSParallelOutboundQueue" />
   ```

2. Using JMSAdmin, define the JMS name for the serial and parallel outbound queues.
3. Add an entry in the VIEWREG table to define the response needed for the inbound request. This step is required both for existing WebSphere Commerce commands and for new commands. Use the following values:

   VIEWNAME
   The same value as for regular HTTP responses

   INTERFACENAME
   `com.ibm.commerce.messaging.viewcommands.MessagingViewCommand`

   CLASSNAME
   `com.ibm.commerce.messaging.viewcommands.MessagingViewCommandImpl`

   PROPERTIES
   `docname=`*`JSP filename`*

   DEVICEFMT_ID
   - -20000 for XML inbound messages
   - -30000 for legacy inbound messages
   - Add -5 for new inbound messages

Chapter 3. Program Adapter Security for MQSeries

To change the level of security, you must change the value of the class attribute in the `SessionContext` node of the MQSeries adapter configuration.

Before the request is executed by the WebController, the WebController determines the credentials the request must execute under. This is determined by the specified class, which uses the CommandProperty object to determine the credentials of the request.

There are two levels of security available:

- Level 1: Limited security.
- Level 2: User ID and password required for every request.

You can also create your own customized security class. The only restriction is that it must implement the `ProgramAdapterSessionContext` interface.

Security Level 1: Limited security

This level of security is enabled by default at installation. It assumes that all requests are to be processed by a default user ID unless otherwise specified. If the request is to be processed by another user ID, this user ID is specified in the request. The password of the specified user ID is not required and the request is processed by WebSphere Commerce using the credentials of the specified user.

The class that implements this security level is `messaging.programadapter.security.DefaultCredentialsProgramAdapterSessionContextImpl`.

Security Level 2: Logon and Password for every request

This level of security requires that a request sent to WebSphere Commerce contains a user ID and password. If the password does not match the specified user ID, a security exception is thrown and the request is rejected. If the user ID and password are not specified in the request, the request is processed as a guest user. This means that commands that need authorization before executing must have the credentials specified in the request.

The class that implements this security level is `messaging.programadapter.security.CredentialsSpecifiedProgramAdapterSessionContextImpl`.

The validation of credentials uses the same technique used when customers interact with WebSphere Commerce through a browser. This takes into consideration whether the user registration is handled by WebSphere Commerce or a third party software.

All of the WebSphere Commerce supported XML integration messages support this level of security, although credential specification is not mandatory.

Program Adapter security for HTTP requests

This level of security requires that a request sent to WebSphere Commerce contains a user ID and password. If the password does not match the specified user ID, a security exception is thrown and the request is rejected. If the user ID and password are not specified in the request, the request is processed as a guest user. This means that commands that need authorization before executing must have the credentials specified in the request. This form of security behaves in a similar manner to HTTP requests made from a browser client.

The class that implements this security level is
`messaging.programadapter.security.CredentialsSpecifiedProgramAdapterSessionContextImpl`.

The validation of credentials uses the same technique used when customers interact with WebSphere Commerce through a browser. This takes into consideration whether the user registration is handled by WebSphere Commerce or a third party software.

All of the WebSphere Commerce supported XML integration messages support this level of security,although credential specification is not mandatory. Legacy messages are not supported by this implementation of security.

Message composition services

Some of the WebSphere Commerce messages use the message composition services. Through the use of JSP templates, the composition services generate a message before it is sent through the transport. If the composition service is used for a message, it runs a JSP passing it information such as an order number or a store number. When the template is executed, the JSP may retrieve any additional information necessary for the message from the database using data beans. The output is generated and the formatted message is sent through the transport. You can modify the message templates just as you would any other JavaServer Page.

Some of the features of the generated message you may want to modify include:

- The layout of the message.
- The information about your store, the order, or the customer that is retrieved from the database and displayed on the page.
- The text of the messages to the customer such as “Thank you for ordering with us”.
- The format of the output generated. For example, you may want to send a message in HTML format, plain-text format, or XML format.

Messaging system

The WebSphere Commerce messaging system gives WebSphere Commerce the ability to communicate with its external environment. This communication includes sending messages to and receiving messages from back-end systems or external systems, as well as sending notification to customers and administrators that events have occurred within WebSphere Commerce. This is accomplished through two subsystems: an inbound system that manages inbound messages coming from back-end and external systems, and an outbound messaging system that allows you to send notification to users as well as outbound messages to back-end systems and external systems.

For example, you can set up the messaging system to send e-mail messages notifying your customers that their orders have been shipped. The messaging system provides a mechanism for integrating WebSphere Commerce with back-end systems. You can configure WebSphere Commerce to send an outbound message to a back-end system whenever an order is created at your store. This order information can be used by the back-end system to do necessary order fulfillment processing. The back-end system can later send order status messages back to WebSphere Commerce indicating that order delivery has taken place or an order invoice has been issued. An e-mail can also be sent to update the customer.

Generic Application and System Error XML messages

In WebSphere Commerce, generic application and system errors can occur. These messages appear in XML format, and are sent to the outbound queue.

A generic application error message is sent to the outbound queue if the error is related to the user. When a user enters an invalid parameter in an XML message, an ECApplicationException is thrown. The message is then sent to the outbound queue and the exception is documented in a log file.

Note: When an exception of this type is thrown, the Web controller will not retry the command, even if it is specified as a command that could be retried.

A generic system error message is sent to the outbound queue if a run-time exception or a WebSphere Commerce configuration error is detected, such as null-pointer exceptions and translation rollback exceptions.

The contents of generic XML messages vary depending on the contents of inbound XML messages however, the format is similar to the following:

```
<?xml version="1.0" encoding="UTF-8"?>

<WCS_Error type="GenericApplicationError"> (or <WCS_Error type="GenericSystemError">)

<RequestAttributes>

...(Error parameters and data of inbound XML message)

<excMsg>(Error message key, e.g. _ERR_REMOTE_EXCEPTION)</excMsg>

...

</RequestAttributes>

</WCS_Error>
```

Setting up outbound message composition

To set up and use the composition service for an outbound message, do the following:

1. Assign the transports to the appropriate message type, using either of these methods:
   - Assigning a transport method to a message type for a site
   - Assigning a transport method to a message type for a store
     A valid device format, as specified in the DEVICEFMT table, must be specified for each transport to be used.
2. Referring to the information in the topic Outbound messaging system interface, create a messaging system object using the SendMsgCmd task command. Use the setMsgType() and setStoreId() initialization services.
3. Invoke the messaging system compose method.

Example of using the messaging system composition service

The following is an example of how you might use the messaging system composition service. If you have a store named DemoStore and you want to assign two transports, e-mail and file, to the OrderAuthorized message type, you would do the following:

1. Add an entry to the VIEWREG table for the JSP file to use for composing this outbound message. The keys for the VIEWREG table are the view name, the store ID, and the device format ID. For more information on how the VIEWREG table is used, refer to the *WebSphere Commerce Programmer's Guide*.

Important: Each view created to be used by the Messaging System's compose service must use the Messaging View Command for the interface and class name fields. It must also contain the name of the JSP file in the docname field. To summarize:

INTERFACENAME
`com.ibm.commerce.messaging.viewcommands.MessagingViewCommand`

CLASSNAME
`com.ibm.commerce.messaging.viewcommands.MessagingViewCommandImpl`

PROPERTIES
Use the following format to point to the JSP file 'docname=*jsp file*'.

DEVICEFMT_ID
Represents the device format and should use the value -3 (the standard device format) unless using custom device formats for your application. The DEVICEFMT_ID specified in the VIEWREG entry must correspond to the device format selected when assigning a message type to a transport.

For example, if the store ID for DemoStore were 5 and the viewname is OrderAuthorized you could insert a record using the following SQL statement:

```
insert into viewreg (VIEWNAME,STOREENT_ID,DEVICEFMT_ID,INTERFACENAME,CLASSNAME,
PROPERTIES)values
('OrderAuthorizedView',5,-3,
'ibm.commerce.messaging.viewcommands.MessagingViewCommand'
'ibm.commerce.messaging.viewcommands.MessagingViewCommandImpl',
'docname=OrderAuthorized.jsp');
```

2. Use the Administration Console to assign the e-mail and file transports to the OrderAuthorized message and configure the settings. This can be done using either site or store level administration authority. Creating settings on site level will make it accessible to all stores.
3. In the implementation of a command, instantiate the SendMsgCmd command to use messaging services and call the setMsgType() and setStoreID() methods, using the message ID of the OrderAuthorized message type and the store ID of DemoStore. If you need to use site-level configuration, specify `0` as the store ID and attach `"&storeDir=no"` at the end of the JSP name. Otherwise, use your store ID. (If no configuration exists for your store, the Messaging System defaults to site-level configuration automatically.)
4. Invoke the compose method of the outbound messaging system interface and pass any additional parameters in the form of a TypedProperty object. By specifying a viewname, you will override the message types default viewname used when composing the message.
5. Call sendImediate or sendTransacted on SendMsgCmd if you want the message to be sent immediately or after the transaction has successfully been committed. Refer to the Messaging System documentation for a further explanation of the use of each method.
6. Call execute method of the SendMsgCmd to execute sending.

Error handling in the messaging system composition service

When an error occurs in the processing of a JavaServer page, the result of the page typically contains extensive information generated by the runtime. If this is not the desired result on a JavaServer page failure, there are two potential approaches to make the behavior more predictable.

In the first approach, you can specify an error page in your JavaServer page, which runs if an unanticipated error is encountered, such as an uncaught exception. To use this approach, you need to include the following line in your main page:

```
<%@ page errorPage="YourError.jsp" %>
```

YourError.jsp:

```
<%@ page isErrorPage="true" %>
```

Note: In the event of an error, the result of the message composition is the result of YourError.jsp. For more details please see the JSP 1.0 specification.

In the second approach, instead of having the error message processed as an outbound message, you can use an error JavaServer page that generates output that is processed as an exception. To do this, use the ComposerError.jsp as the basis for handling the error. Do this the same way as described above, but the beginning of YourError.jsp should start with the first two lines of ComposerError.jsp. This allows the composition runtime to detect that an error occurred, and to raise an exception. The ComposerError.jsp can be found in the following directory:

2000

drive:\Program Files\WebSphere\AppServer\installedApps\WC_Enterprise_App_<*instance_name*>.ear\wcstores.war

NT

drive:\WebSphere\AppServer\installedApps\WC_Enterprise_App_<*instance_name*>.ear\wcstores.war

AIX

/usr/WebSphere/AppServer/installedApps/WC_Enterprise_App_<*instance_name*>.ear/wcstores.war

Solaris

/opt/WebSphere/AppServer/installedApps/WC_Enterprise_App_<*instance_name*>.ear/wcstores.war

400

/QIBM/Userdata/WebASAdv4/<*WAS_instance_name*>/installedApps/WC_Enterprise_App_<*instance_name*>.ear/wcstores.wa

ComposerError.jsp are as follows:

```
ERROR
<%@ page isErrorPage="true" %>
```

errorPage="ComposerError.jsp" in the main JSP page. If you want extra information, you can copy this file to another filename, such as YourError.jsp, and place the extra error information after these two lines in the new file. You would then set errorPage="YourError.jsp" in the main JSP page. Any extra output specified after these two first lines will be part of the text of the exception thrown by the composition service.

Invoke the messaging system compose method

To invoke the compose() method of the outbound messaging system interface, specify the following parameters:

- `viewname`: The name of the composition view to be used, as specified in the VIEWNAME column of an existing record in the VIEWREG table. For more information on how the VIEWREG table is used, refer to the *WebSphere Commerce Programmer's Guide*.

  Important: In the record referred to in the VIEWREG table, the values of the INTERFACENAME and CLASSNAME columns must contain the name of the interface and class associated with all WebSphere Commerce messaging system view commands. The name of the interface must be `com.ibm.commerce.messaging.viewcommands.MessagingViewCommand`. The name of the class must be `com.ibm.commerce.messaging.viewcommands.MessagingViewCommandImpl`.
- `CommandContext`: For information on the CommandContext interface or the CommandContextImpl class that implements the interface, refer to the *WebSphere Commerce Programmer's Guide*.
- `TypedProperty`: The values in the typed property must be strings, or objects that implement the toString() method. For more information on the TypedProperty, refer to the *WebSphere Commerce Programmer's Guide.*

The compose() method runs a view command for each of the transports enabled and assigned to the current message type in the Administration Console. The method performs the following processes:

- It uses the `viewname` parameter as well as the `storeId` and device format from each transport, as defined in the Administration Console. These values are used to look up the view command in the VIEWREG table.
- It runs the view command, passing it the values specified in the `TypedProperty` parameter. When the command is run, the system uses the viewname, storeId, and device format id to look up the JSP template in the PROPERTIES column of the VIEWREG table. The JSP template is run and passed the values in the `TypedProperty` parameter.
- The JSP composes the message, and it is sent through the appropriate transport when a send method is invoked in the object. Sending may be done using transacted, immediate, or request-reply on the messaging object on which composition was run.

Chapter 4. OrderItemStatus command

The OrderItemStatus command is called internally by the OrderStatus, OrderConfirmStatus, OrderInvoiceStatus, and OrderShippingStatus commands when an Update_WCS_OrderStatus, Update_NC_OrderStatus, or Order Status Update message is received from a back-end system. The command updates information regarding the status of an an existing order for an item.

Parameters

The following is a list of the parameters for the command. Each parameter corresponds to a field in the ORDISTAT table:

versioning: String "TRUE" or "FALSE". optional, Default is FALSE.
If TRUE, versioning will be enabled.

orderItemId: Integer
WebSphere Commerce order item reference number as defined in the ORDERITMES_ID in table ORDISTAT. This is a foreign key that references column ORDERITEMS_ID in table ORDERITEMS.

merchantItemNumber: String
Order item number generated in the backend system as defined in the OIMITEM column.

PartNumber: String
Item Product number/SKU as defined in the PARTNUMBER column.

UnitOfMeasure: String
Item unit of measure as defined in the OIUOFM column.

RequestQuantity: Integer
Quantity of items requested as defined in the OIQTREQUEST column.

ConfirmQuantity: Integer
Quantity of items confirmed as defined in the OIQTCONFIRM column.

ShipQuantity: Integer
Quantity of items shipped as defined in the OIQTSHIP column.

Currency: String
The ISO 4217 currency type in which the price is expressed as defined in the OICPCUR column.

UnitPrice: BigDecimal(20,4)
The unit price for the product as defined in the OIUNPRC column.

PriceTotal: BigDecimal(20,4)
The total product price for the item as defined in the OIPRTOT column.

TaxTotal: BigDecimal(20,4)
The total tax for the item as defined in the OITXTOT column.

ShippingTotal: BigDecimal(20,4)
The total shipping charge for the item as defined in the OISHTOT column.

ShippingTaxTotal: BigDecimal(20,4)
The total tax on shipping charge for the item as defined in the OISHTXTOT column.

Status: String
The status of the item as defined in the OISTATUS column.

PlaceDateTime: Timestamp
The date that the item is actually placed as defined in the OIPLTIME column.

RequestShipDateTime: Timestamp
The date that the item is requested to be shipped as defined in the OIRSTIME column.

ScheduleShipDateTime: Timestamp
The date that the item is scheduled to be shipped as defined in the OISSTIME column.

ActualShipDateTime: Timestamp
The date that the item is actually shipped as defined in the OIASTIME column.

InvoiceDateTime: Timestamp
The date that the item is invoiced as defined in the OIINVTIME column.

InvoiceValue: BigDecimal(20,4)
The net value that the item is invoiced as defined in the OIINVVAL column.

itemShipCondition: String
Code to be designate whether partial shipment of the item will be accepted as defined in the OISCOND column. 'SC': Ship Complete, 'SP': Ship Partial.

itemComment: String
Comments regarding the item status as defined in the OICMNT column.

field1: Integer
Item status customization field 1 as defined in the FIELD1 column.

field2: BigDecimal(15,2)
Item status customization field 2 as defined in the FIELD2 column.

field3: String
Item status customization field 3 as defined in the FIELD3 column.

Behavior

- The first order item status for an order must have orderItemId and merchantItemNumber specified.
- Check if the order item specified in orderItemId exists in the ORDERITEMS table.
- In each subsequent order item status which already has an entry in the ORDISTAT table, the orderItemId is not required to allow a new line item to be created by the back-end system for existing line item.
- Create or update a row in the ORDISTAT table using information provided.
- If versioning is enabled and this is the first order item status for the order, a new row will be created in the ORDISTAT table. Otherwise a copy of the last order item status will be made and its version (OIVERNBR) will be set to the maximum number of existing versions +1. A new row will be created containing all of the information provided, with its version (OIVERNBR) set to 0.

Exception Conditions

- The orderItemId specified is not a valid order number in the ORDERITEMS table.
- Either orderItemId or merchantItemNumber is not specified in the first order item status for a particular order item.
- The orderItemId and merchantItemNumber do not match the one already in ORDISTAT table for a particular order item in a subsequent order item status update.

GetPickPackListDetail command

This command retrieves pick ticket and packing list information for the Inquire_WCS_PickPackListDetail XML message.

Parameters

pickBatchId
The reference number of the pick batch as defined in the PICKBATCH_ID column of the PICKBATCH table.

Behavior

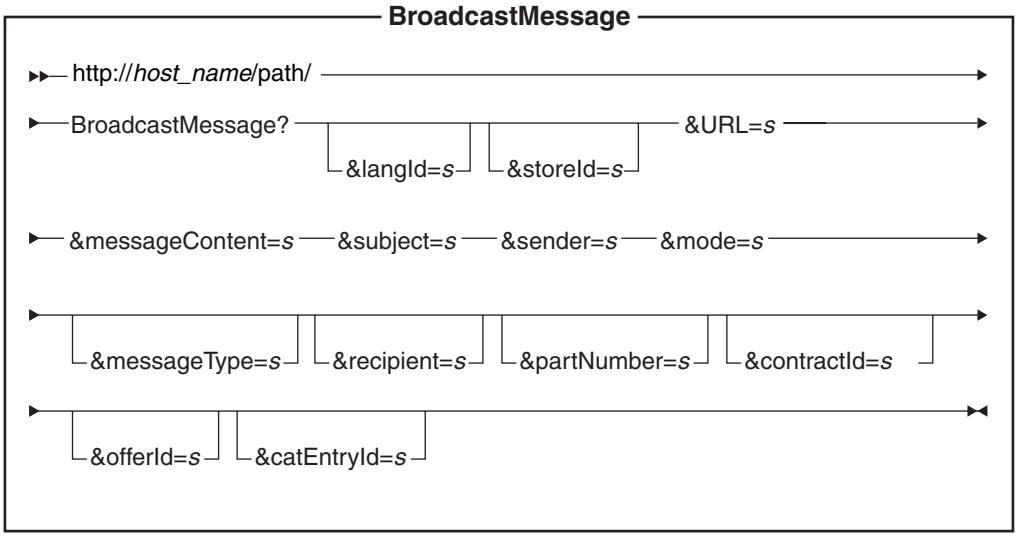
- Checks if the pickBatchId exists in the PICKBATCH table.
- Checks that there is at least one record in the ORDRELEASE table for that pickBatchId.
- Retrieves the pick ticket XML from the PICKBATCH table and the packing list XML for the given pickBatchId and forwards to the view task to compose the Report_WCS_PickPackListDetail message.
- The Report_WCS_PickPackListDetail message containing pick ticket and packing list information is sent in response.

Exception Conditions

- The pickBatchId is invalid or not found in the PICKBATCH or ORDRELEASE tables.

BroadcastMessage command

This command broadcasts an e-mail to customers.



Parameter Values

http://*host_name/path/*
The fully qualified name of your WebSphere Commerce Server and the configuration path.

langId Sets or resets the preferred language for the duration of the session; the supported languages for a store are found in the STORELANG table.

storeId
The reference number of the store from which the message is being sent.

URL The URL to be called when the command completes successfully.

messageContent
The content of the message being sent. If specified, this content overrides the content of the message template. This parameter must be specified if the `messageType` parameter is specified.

subject
The content of the Subject line in the e-mail. If the subject is not specified, the default subject specified in the message type configuration will be used.

sender
The content of the From line in the e-mail. If the sender is not specified, the default sender specified in the message type configuration is used.

mode Possible values:

1. Sends the message to all shoppers for a given store who have purchased any product and have indicated that they would like to receive promotional e-mail.
2. Sends the message to all shoppers for a given store who have purchased the indicated SKU and have indicated that they would like to receive promotional e-mail.
3. Sends the message to all shoppers who have indicated that they would like to receive promotional e-mail.

messageType
Reference to a predefined message template defined in the MSGTYPE_ID column of the MSGTYPES table. If messageType is not specified, the site-level message type for broadcast messaging will be used. By default, the site-level message type has no associated message template, so it is recommended that stores have their own broadcast message type set up. See Message logging for information on how to set up new message types.

recipient
The extra recipients of the e-mail, in a comma-separated list.

partNumber
The part number of the item associated with the e-mail in mode 2.

contractId
Target members that have purchased under this contract in mode 2.

offerId
Target members that have purchased under this offer in mode 2.

catEntryId
Target members that have purchased this catalog entry in mode 2.

Behavior

- Calls the GetRecipientsCmd task command to create a list of broadcast e-mail message recipients.
- Calls another task command to send e-mail messages by providing the return e-mail address, subject of message, the message, and the comma-separated list of recipients parameters.
- Uses the template corresponding to the specified message type for the body of the message.

Exception Conditions
If the list of recipients could not be determined or there is an error delivering the message, the BroadcastMessageErrorView error task is called.

OrderInvoiceStatus command

The OrderInvoiceStatus command is used when an Update_WCS_OrderStatus XML message with an OrderStatusType tag value of `OrderInvoice` is received from a back-end system. The command updates information regarding the invoice status of an an existing order. It has the same parameters, behavior, and exception conditions as the OrderStatus command. The only difference between the two commands is that the OrderInvoiceStatus command sets the value of the orderStatus parameter to a default value of `I` which means the order has been invoiced if it is not provided in the message.

For detail on the parameters, behavior, and exceptions handling for the OrderInvoiceStatus command, refer to the OrderStatus command.

OrderShippingStatus command

The OrderShippingStatus command is used when an Update_WCS_OrderStatus XML message with an OrderStatusType tag value of `OrderShipping` is received from a back-end system. The command updates information regarding the shipping status of an an existing order. It has the same parameters, behavior, and exception conditions as the OrderStatus command. The only difference between the two commands is that the OrderShippingStatus command sets the value of the orderStatus parameter to a default value of `S` which means the order has been shipped if it is not provided in the message.

For detail on the parameters, behavior, and exceptions handling for the OrderShippingStatus command, refer to the OrderStatus command.

OrderStatus command

The OrderStatus command is used when an Update_WCS_OrderStatus, Update_NC_OrderStatus, or Order Status Update message is received from a back-end system. The command updates information regarding the status of an an existing order.

Parameters

sequenceNumber
String. WCS order status sequenceNumber for serialization.

lastUpdateTimestamp:
String in timestamp format. WebSphere Commerce order status last update timestamp for serialization.

versioning: String “TRUE” or “FALSE”. Default is FALSE.
If TRUE, the versioning will be enabled.

The parameters listed below correspond to columns in the ORDSTAT table.

orderId: Integer.
WebSphere Commerce order reference number as defined in the ORDERS_ID column in table ORDSTAT. This is a foreign key that references column ORDERS_ID in the ORDERS table.

merchantOrderNumber: String, mandatory.
Order number generated in the backend system as defined in the OSMORDER column.

currency: String
The ISO 4217 currency type in which the price is expressed as defined in the OSCPCUR column.

priceTotal: BigDecimal(20,4)
The total product price for the order as defined in the OSPRTOT column.

taxTotal: BigDecimal(20,4)
The total tax for the order as defined in the OSTXTOT column.

shippingTotal: BigDecimal(20,4)
The total shipping charge for the order as defined in the OSSHTOT column.

shipingTaxTotal: BigDecimal(20,4)
The total tax on shipping charge for the order as defined in the OSSHTXTOT column.

orderStatus: String
The status of the order as defined in the OSSTATUS column.

placeDateTime: Timestamp
The date that the order is actually placed as defined in the OSPLTIME column.

requestShipDateTime: Timestamp
The date that the order is requested to be shipped as defined in the OSRSTIME column.

scheduleShipDateTime: Timestamp
The date that the order is scheduled to be shipped as defined in the OSSSTIME column.

actualShipDateTime: Timestamp
The date that the order is actually shipped as defined in the OSASTIME column.

invoiceDateTime: Timestamp
The date that the order is invoiced as defined in the OSINVTIME column.

invoiceValue: BigDecimal(20,4)
The net value that the order is invoiced as defined in the OSINVVAL column.

shipCondition: String
Code to be designate whether partial shipment of the order will be accepted as defined in the OSSCOND column. 'SC': Ship Complete, 'SP': Ship Partial.

shippingModeFlag: String
Code to indicate shipping address and shipping mode are at order level or order item level as defined in the OSSMFLAG column. 'O': Order level; 'I': Item level.

comment: String
Comments regarding the order status as defined in the OSCMNT column.

field1: Integer
Order status customization field 1 as defined in the FIELD1 column.

field2: BigDecimal(15,2)
Order status customization field 2 as defined in the FIELD2 column.

field3: String
Order status customization field 3 as defined in the FIELD3 column.

items: Vector of Hash table.
Each hash table represents the parameters for one item.

Behavior

- The first order status for an order must have orderId and merchantOrderNumber specified.
- Check if the order specified in orderId exists in the ORDERS table.
- If serialization information is available, such as the sequenceNumber and the lastUpdateTimestamp parameters, check if the values are more recent than the ones already in the ORDSTAT table, if not then no update will be performed.
- Create or update a row in the table ORDSTAT using all the provided information.
- If versioning is enabled and this is the first order status for the order a new row will be created in ORDSTAT table, otherwise a copy of the last order status will be made and its version (OSVERNBR) will be set to the maximum number of existing version +1. A new row will be created containing all provided information with its OSVERNBR
  value set to 0.
- Update the value in the STATUS column of table ORDERS to 'G'.

Exception Conditions

- The orderId specified is not a valid order number in the ORDERS table.
- Either the orderId or merchantOrderNumber is not specified in the first order status for a particular order.
- The orderId and merchantOrderNumber do not match one already in ORDSTAT table for a particular order in a subsequent order status udpate.
- Serialization information is provided and the order status message is out of sequence

ProductOfferPriceUpdate command

The `ProductOfferPriceUpdate` command is used to update product price information for the Update_WCS_ProductPrice, Update_NC_ProductPrice, and Product Price Update messages.

Parameters

offerId
The reference number that identifies the offer.

partNumber
The part number of the catalog entry as defined in the PARTNUMBER column of CATENTRY table.

memberId
The reference number that identifies the owner of the catalog entry.

catEntryId
The catalog entry offered for sale.

currency
The ISO 4217 currency type in which the price is expressed. This value is mandatory.

offerPrice
The offer price to be used for this update.

precedence
The precedence to be used for this update.

tradingPositionContainerId
The TradingPositionContainer of which the offer is a part.

startDateTime
The start of the time range during which the Offer is effective.

endDateTime
The end of the time range during which the Offer is effective.

minimumQuantity
The minimum quantity that can be purchased in a single order under this offer.

maximumQuantity
The maximum quantity that can be purchased in a single order under this offer.

quantityUnit
The unit of measure for minimumQuantity and maximumQuantity.

Behavior

- The command updates a record in the OFFERPRICE table.
- The process by which the primary key is determined varies slightly, depending on the format of the inbound message that executed the command. For a description of how the primary key is determined, refer to the specific inbound message.

Exception Conditions

- The currency parameter is empty.
- The offerId parameter value can not be found in the OFFERPRICE table.
- The catalog entry cannot be found using the memberId, which owns the store along with the partNumber.
- The tradingPositionContainerId parameter value cannot be found in the OFFER table for the matching offerId.
- The precedence parameter value exceeds the maximum value. The precedence must be less than 10^{16}.

ProductInventoryUpdate command

The `ProductInventoryUpdate` command is used to update product inventory for the Update_NC_ProductInventory message, Update_WCS_ProductInventory message, or Product Quantity Update message.

Parameters

catEntryId
WebSphere Commerce catalog entry as defined in the CATENTRY_ID column in INVENTORY table. If the catalog entry is empty, then the combination of the part number and the member id which owns both the store and the catalog will be used to get the catalog entry.

partNumber
The part number of the catalog entry as defined in the PARTNUMBER column of CATENTRY table. Together with the member id which owns the catalog, it is used to get the key catEntryId in the table CATENTRY, if the catEntryId parameter is empty.

storeId
The store id that references column STORE_ID in the table INVENTORY. Together with catalog entry and the default fulfillment center, this is a key to the row in the table INVENTORY.

inventoryQuantity
The quantity as defined in the QUANTITY column in table INVENTORY.

fulfillmentCenterID
The fulfillment center id that references column FFMCENTER_ID in the table INVENTORY. This a key to the FFMCENTER database table.

Behavior

- The command updates a record in the INVENTORY table.
- The store id (which references STORE_ID in the INVENTORY table) is mandatory.
- The catalogEntryId and the storeId are used to update a row in the INVENTORY table.
- If the catEntryId is not present, then the storeId (STORE_ID) is used to obtain the member id (which references MEMBER_ID in the CATENTRY table). The member id must be the same as the owner of the catalog. The member id (MEMBER_ID), along with the partNumber (which references PARTNUMBER in the CATENTRY table) are used to obtain the catEntryId. The catEntryId, along with the store id and the default fulfillment center id for that catalog entry are used to update a row in the INVENTORY table.
- If the row in the table INVENTORY does not exist, an error will occur.

Exception Conditions

The command generates an entry in the error log if the following exceptions are encountered:

- The storeId does not exist.
- The catEntryId can not be found using the member id which owns the store along with the partNumber.
- The catEntryId, along with storeId and default fulfillment center id, can not find a matching row in table INVENTORY.

SendWCSOrder task command

The SendWCSOrder command is used by the WebSphere Commerce system to send the Order Create legacy message to back-end systems.

Behavior

- The task command is enabled by assigning it to the OrderMessagingCmd interface within the OrderProcess command. Once enabled, it is called before the OrderProcess command finishes processing.
- Using the order reference number as its input parameter, it collects all of the necessary order information.
- Based on the available order information, it then builds the Order Create legacy message as a String and then store it in the message using the outbound messaging system content setting services.
- If the message creation is successful, the command attempts to send the message using the outbound messaging system sending services.

Exception Conditions

The command generates an entry in the error log if an exception is encountered.

SendXMLOrder command

The SendXMLOrder command is used by the WebSphere Commerce outbound messaging system to send the Report_NC_PurchaseOrder XML message to back-end systems. The command uses a message composition template to generate the XML message, then the outbound messaging system sends it to a back-end system.

Behavior

- The task command is enabled by assigning it to the OrderMessagingCmd interface within OrderProcess command.
- Once enabled, it is called before OrderProcess command finish processing.
- The task command calls the messaging system composition services, which uses the OrderCreateXML.jsp composition template to collect of the necessary order information and build the Report_NC_PurchaseOrder outbound XML message.
- If composition is successful, the command attempts to send the message using the outbound messaging system sending services.

Exception Conditions

The command generates an entry in the error log if an exception is encountered.

NewInboundMessage command

The `NewInboundMessage` command is used for customized inbound messages that are not implemented using the `user_template.xml` inbound message template definition file. The command is run when the adapter does not recognize an inbound message. This means that it is neither a legacy message nor an XML message defined in the `sys_template.xml` or `user_template.xml` inbound message template definition files. Initially, the NewInboundMessage command contains no programming statements, so you must customize the command yourself.

Behavior

- Stores the inbound message in a String buffer which can be retrieved using the getMessage() method.

Exceptions

If the performExecute() method is not implemented, it generates an exception by default.

OrderConfirmStatus command

The OrderConfirmStatus command is used when an Update_WCS_OrderStatus XML message with an OrderStatusType tag value of `OrderConfirm` is received from a back-end system. The command updates information regarding the confirmation status of an an existing order. It has the same parameters, behavior, and exception conditions as the OrderStatus command. The only difference between the two commands is that the OrderConfirmStatus command sets the value of the orderStatus parameter to a default value of `C` which means the order has been confirmed if it is not provided in the message.

For detail on the parameters, behavior, and exceptions handling for the OrderConfirmStatus command, refer to the OrderStatus command.

ProductListPriceUpdate command

The ProductListPriceUpdate command is used to update product price information for the Update_WCS_ProductPrice XML message. The command is used to update information in the WebSphere Commerce database regarding the listed price of a product.

Parameters

partNumber
The part number of the catalog entry as defined in the PARTNUMBER column of CATENTRY table.

memberId
The reference number that identifies the owner of the catalog entry.

catEntryId
The catalog entry offered for sale.

currency
The ISO 4217 currency type in which the price is expressed. This value is mandatory.

listPrice
The product list price used to update the LISTPRICE table.

Behavior

- The command updates a record in the LISTPRICE table.
- The partNumber, together with memberId are used to get the catalog entry key (CATENTRY_ID in table CATENTRY).
- If the catalog entry matches an existing one in the LISTPRICE table, but the currency type does not match a currency type for any record for that catalog entry, a new record is created in the LISTPRICE table. This allows you to specify prices in different currencies for the same catalog entry.

Exception Conditions

- The currency parameter is empty.
- The catalog entry cannot be found using the memberId, which owns the store along with the partNumber.

Chapter 5. How the outbound messaging system works

The messaging system uses a plug-in model that implements the Common Connector Framework (CCF) to provide a common interface between the system and the various transports. During administration of the system site and store administrators can perform the following tasks:

- Add, enable, and configure transports. The administrator creates the settings using the Administration Console. Communication between the messaging system and the transports take place through a singular administration interface.
- Maintain profiles, assigning transports to individual message types and indicating the settings to be used for each one. The administrator does this using the Administration Console.

At run time, when a message is generated by a WebSphere Commerce sub-system, the following events occur:

1. The appropriate profile is retrieved for the message type. If no store profile exists for that message, the site profile is used. The profile is used to determine what transport method and settings are used.
2. If the message uses the composition service, a template is used to generate the message.
3. The message is sent through the run time interface to the transport, which delivers the notification.

The use of a common interface with external transports allows the implementation details of the transport to be kept separate from the operation of the messaging system. This architecture makes it possible to plug in additional transports that adhere to the CCF interface.

Outbound messaging system

The WebSphere Commerce messaging system allows you to manage all aspects of defining and sending messages generated within WebSphere Commerce. It allows you to control the manner in which administrators, customers, back-end and fulfillment center systems are notified of various events, such as customer orders or system errors.

To configure the outbound messaging system use the Administration Console. The messaging system can send messages using transports such as e-mail using SMTP and file using UTF-8 encoding. For e-mail the supported outbound protocol is SMTP, the message encoding depends on the specified language. Optionally, you can configure the messaging system to send messages to a back-end or fulfillment center system using MQSeries.

The runtime environment of the outbound messaging system provides a highly customizable messaging environment. These features include the following:

- Composition services
  Customize messages using predefined JSP templates.
- Multiple message transmissions support
  Allows you to send a single message through more than one transport.
- Multiple notification messages over the same transport
  This is useful for sending broadcast emails to multiple recipients.
- Support for three processing types:
  - Transacted
    Use for messages that should be sent upon successful completion of the current transaction.
  - Immediate
    Use for messages that should be sent when the event takes place in WebSphere Commerce. The message is sent whether the transaction commits or not.

– Request-reply
  Use for messages that require a response message from the back-end system.

Note: Ensure transport attributes, for example e-mail addresses, and file locations are valid. The Messaging System does not validate attributes; incorrect attributes will result in failure to send the message.

Outbound messaging system administration

Administration of the outbound messaging system can be divided into two main categories: site administration and store administration. A Site Administrator provides the basic framework to be used by all stores within the site, such as which transports can be used. For example, a store may not use e-mail as a transport unless it has previously been configured and enabled by the Site Administrator. Store administrators can then accept the settings made at the site level, or they can modify them for their store.

Outbound back-end integration messages

An outbound back-end message is a WebSphere Commerce-generated request that can be sent to an external system. WebSphere Commerce can be configured to generate the Report_NC_PurchaseOrder XML outbound message that allows you to communicate to back-end systems that an order has been placed. The XML message is generated and sent out by the outbound messaging system, encoded in Unicode UTF-8 format. You can also use the legacy Order Create message, which performs a similar function.

The outbound messages contain order information sent from the WebSphere Commerce Server to external systems, where further order fulfillment processes take place. To enable the outbound message, first you need to choose which one you want to use, either the Report_NC_PurchaseOrder XML message or the legacy Order Create message. The two can not be enabled at the same time.

If you choose to use the Report_NC_PurchaseOrder XML message, update your CMDREG database table using the following SQL statement:

```
update cmdreg set
classname='com.ibm.commerce.messaging.commands.SendXMLOrderCmdImpl' where
interfacename='com.ibm.commerce.order.commands.OrderMessagingCmd'
```

This assigns the SendXMLOrderCmdImpl task command which generates and sends the message to the OrderMessagingCmd interface of the OrderProcess command.

If you choose to use the legacy Order Create message, update your database CMDREG table using the following SQL statement:

```
update cmdreg set
classname='com.ibm.commerce.messaging.commands.SendWCSOrderCmdImpl' where
interfacename='com.ibm.commerce.order.commands.OrderMessagingCmd'
```

This assigns the SendWCSOrderCmdImpl task command which generates and sends the message to the OrderMessagingCmd interface of the OrderProcess command. You need to restart your WebSphere Commerce instance in order for the change to take effect.

You can also create new outbound back-end integration messages.

Outbound fulfillment integration messages

An outbound fulfillment integration message is a WebSphere Commerce generated request that can be sent to a fulfillment center system. WebSphere Commerce can be configured to generate outbound messages in response to inbound messages, containing information to be communicated to fulfillment center systems.

The outbound XML messages are sent out by the outbound messaging system encoded in Unicode UTF-8 format. Refer to Message composition templates for information on the JSP files that generate the following outbound messages.

The Response_WCS_ExpectedInvRecord outbound message allows you to respond to fulfillment center systems when a request for an expected inventory record has been created. It contains the RA_ID and RADETAIL_ID that are generated. The outbound message is invoked by the Create_WCS_ExpectedInventoryRecord message and generated by the RACreateResult.jsp file.

The Response_WCS_PickBatch outbound message allows you to communicate to fulfillment center systems that a pickbatch has been created. The outbound message is invoked by the Create_WCS_PickBatch message and generated by the PickBatchResult.jsp file.

The Report_WCS_PickPackListDetail outbound message is invoked by the Inquire_WCS_PickPackListDetail message and allows you to respond to a fulfillment center request by sending details for a specific PICKBATCH_ID. It contains the pick ticket and packing list from the ORDRELEASE table for the given PICKBATCH_ID.

The Response_WCS_CreateInvReceipt outbound message allows you to communicate to fulfillment center systems that an inventory receipt has been created. The outbound message is invoked by the Create_WCS_InventoryReceipt message and generated by the CreateInvReceiptOK.jsp file.

The Response_WCS_UpdateInvReceipt outbound message allows you to respond to fulfillment center systems informing them that inventory has been updated upon receipt. The outbound message is invoked by the Update_WCS_InventoryReceipt message and generated by the UpdateInvReceiptOK.jsp file.

The Response_WCS_CreateShipConfirm outbound message is invoked by the Create_WCS_ShipmentConfirmation message and allows you to respond to fulfillment center systems confirming that a shipment confirmation has been created.

The Release_WCS_ShipmentNotify outbound message is an outbound e-mail message that allows you the option of notifying customers when an order has been shipped.

Outbound messaging system store administration

The Store Administrator is responsible for enabling the transport methods that are used by the store. The Store Administrator can add, activate, deactivate, and configure transport methods for the store, and assign transport methods to message types. The Store Administrator has the option to either accept the settings created by the Site Administrator, or override them. The following is a list of the tasks involved in store administration:

- Add transport methods.
- Activate or deactivate transport methods.
- Configure transport methods.
- Assign transport methods to message types.

Once a Store Administrator has overridden a site-level setting, any future changes made by the Site Administrator to that particular setting will not affect that store. Changes made to other settings at the site-level, that have not been modified by the Store Administrator, still apply. For example, if e-mail is configured for the site on a SMTP host, `smtp.host1.com`, but Store A has specified `smtp.host2.com`, any future changes site-level to e-mail will not affect the settings for e-mail at Store A.

Outbound messaging system site administration

The Site Administrator can determine which transports are to be supported by the site, and configure them on a site-wide basis. Site level administration provides default settings which can be overridden by store level administration settings.

A Site Administrator can activate and configure transports and message types for the site, or allow Store Administrators to specify their own settings. The following is a list of the tasks involved in site administration:

- Add transports.
- Activate or deactivate transports.
- Configure transports. This supplies default configurations that a Store Administrator can override.
- Assign transports to message types. These assignments can be overridden by Store Administrators.

Only Site Administrator can perform these tasks:

- Enable error notification to send e-mail messages to administrators
- Enable the MQSeries JMS transport to send messages to back-end system
- Enable order status notification to update customers or administrators on the status of existing orders

Adding new messages to the Transport Adapter

WebSphere Commerce allows you to extend the Transport Adapter to process additional messages. This involves creating an XML template of the message, and mapping the message to a Controller Command. To do this, you will need to have an advanced knowledge of XML, and of the WebSphere Commerce Controller Commands. If your message requires you to create a new Controller Command, you will also need to have an advanced knowledge of the WebSphere Commerce database schema, and Java programming.

Chapter 6. Enabling outbound messaging Send/Receive sending services

WebSphere Commerce can interact with other systems through the outbound messaging Send/Receive sending services. This allows WebSphere Commerce to send a message to another system and wait for a reply. The behavior of Send/Receive is similar to the `SendImmediate` method except that it waits for a return reply from the system receiving its message request.

After sending the request message, WebSphere Commerce listens to the inbound queue and waits until the reply message is placed on the queue with a correlation ID equal to the message ID of the request message.

You should use separate queues for Send/Receive from the queues used for the MQSeries adapter.

To enable WebSphere Commerce to use the Send/Receive message service, do the following:

1. Create a message type in the MSGTYPES table for the new message used for the Send/Receive. The value in the VIEWNAME column is the VIEWNAME used to generate the message to send.
2. Create a command that uses the `sendReceiveImmediate` sending mode and the message type ID created in step 1. This is used to call the SendMsg interface.
3. To generate the outbound message, create an entry in the VIEWREG table to associate the VIEWNAME created in step 1 with a JSP file. The entry must use the following values:

   INTERFACENAME
   `com.ibm.commerce.messaging.viewcommands.MessagingViewCommand`

   CLASSNAME
   `com.ibm.commerce.messaging.viewcommands.MessagingViewCommandImpl`

   PROPERTIES
   `docname=`*`JSP filename`*
4. Create a JSP file to generate the message to send to the outbound queue.
5. Use the Administration Console to ensure that the transport assigned to the Send/Receive is active.
6. Use the Administration Console to define each message type that you created in step 1. In the **Mode** field specify 0. This indicates a Send/Receive mode of communication.

Enabling the messaging system transport adapter

To enable the messaging system transport adapter, do the following:

1. Launch the Configuration Manager.
2. Select ***Host name*** —> **Instance**, then open the **Components** folder.
3. Select **TransportAdapter**.
4. Ensure that the check-box next to **Enable Component** is activated and click **Apply**.
5. Exit Configuration Manager.
6. Restart the WebSphere Application Server.

Assigning the error condition message type to a transport

Ensure that error messages are assigned to the site, not individual stores. To assign the error condition message type to a transport method, do the following:

1. Open the Administration Console and log on as a Site Administrator.
2. From the **Configuration** menu, select **Message Types**. The Message Type Configuration page displays.

3. Click **New**. The Message Transport Assignment page opens.
   a. On the **Message Type** drop-down list select Description of an Error Condition occurring in WebSphere Commerce.
   b. On the **Message Severity** field specify 0 0.
   c. From the **Transport** drop-down list, select E-mail.
   d. From the **Device Format** drop-down list, select Standard Device Format.
4. Click **Next**. The Message Transport Assignment parameter page displays.
   a. On the **Host** field, type the fully qualified name of your mail server.
   b. On the **Protocol** field, type `smtp`.
   c. On the **Recipient** field, specify the administrator who should receive error notification messages. If multiple recipients are specified, separate recipient names with commas.
   d. On the **Sender** field, specify the sender of the message.
   This text appears in the Sender field of the e-mail message.
   e. On the **Subject** field, specify the subject of the message.
   This text appears in the Subject field of the e-mail message.
5. Click **Finish**.

Activating a transport method for a store

To activate an existing transport method for your store, do the following:

1. Open the Administration Console and log on as a Store Administrator.
2. From the **Configuration** menu, select **Transports**. The Transport Configuration page displays.
3. Click the check box next to the transport you want to activate.
4. Click **Change Status**. The page reloads and the status changes.

Adding a transport method to a store

To add a new transport method to your store, do the following:

1. Open the Administration Console and log on as a Store Administrator.
2. From the **Configuration** menu, select **Transports**. The Transport Configuration page displays.
3. Click **Add**. The Add Transport dialog opens.
4. Select the check box next to the transport you want to add to the store. You can select all transports by selecting the check box at the top-left. If there are no transports available, then you have already added all of the transports made available by the Site Administrator.
5. Click **Add** to add the transport, or click **Cancel** to return to the Transport Configuration page.

Configuring a transport method for a store

To configure a transport method for your store, do the following:

1. Open the Administration Console and log on as a Store Administrator.
2. From the **Configuration** menu, select **Transports**. The Transport Configuration page displays.
3. Select the check box beside the transport you want to configure.
4. Click **Configure**. The Transport Configuration Parameters page opens. The name of the transport you selected displays at the top-left of the parameter table.
5. Provide the information for the transport in the appropriate fields.
6. Click **OK** to accept the changes, or click **Cancel** to return to the Transport Configuration menu without making changes.

Note: Do not click **OK** unless you want the settings to take effect. Once you have clicked **OK**, changes made to the configuration of this transport by the Site Administrator will no longer affect this store. If you have not made any changes, or do not want the settings to take effect , click **Cancel**.

Deactivating a transport method for a store

To deactivate an existing transport method for your store, do the following:

1. Open the Administration Console and log on as a Store Administrator.
2. From the **Configuration** menu, select **Transports**. The Transport Configuration page displays.
3. Click the check box for the transport you want to deactivate.
4. Click **Change Status**. The page reloads, indicating that the transport status is now Inactive.

Assigning a transport method to a message type for a store

A Store Administrator can accept the transport method assignments made by the Site Administrator, or override them for their store. To assign transport methods to be used for specific message types for a store, do the following:

1. Open the Administration Console and log on as a Store Administrator.
2. From the **Configuration** menu, select **Message Types**. The Message Type Configuration page displays.
3. Click the check box next to the message type to which you want to assign a transport, and click **Change**. If the message type is not in the list, click **New**. The Message Transport Assignment page opens.
4. If this is a new transport assignment, select the message type to which a transport is to be assigned from the **Message Type** drop-down list.
5. Type the transport configuration values in the appropriate fields. In general, a **message severity** of 0,0, and a Standard Device Format is recommended.
6. Click **Next** to configure the transport parameters for the specified message type.
7. Type the attributes for the transport you have chosen for this message type.
8. Click **Finish** to save your changes or **Cancel** to return to the Message Type Configuration page.

Note: Do not click **Finish** unless you want the settings to take effect. Once you have clicked **Finish**, changes made to the configuration of this transport by the Site Administrator will no longer affect this store. If you have not made any changes, or do not want the settings to take effect effect, click **Cancel**.

Activating a transport method for a site

To activate an existing transport method for your site, do the following:

1. Open the Administration Console and log on as a Site Administrator.
2. From the **Configuration** menu, select **Transports**. The Transport Configuration page displays.
3. Click the check box next to the transport you want to activate.
4. Click **Change Status**. The page reloads, indicating that the status of the transport is now Active.

Adding a transport method to a site

To add a new transport method to your site, do the following:

1. Open the Administration Console and log on as a Site Administrator.
2. From the **Configuration** menu, select **Transports**. The Transport Configuration page displays.
3. Click **Add** to open the Add Transport page.
4. Select the check box next to the transport you wish to add to the site. You can select all transports by selecting the top check-box.

5. Click **Add** to accept the changes, or click **Cancel** to return to the Transport Configuration page. When you add a transport method to a site, it is automatically activated.

Assigning a transport method to a message type for a site

To assign a transport method to be used for specific message types for a site, do the following:

1. Open the Administration Console and log on as a Site Administrator.
2. From the **Configuration** menu, select **Message Types**. The Message Type Configuration page displays.
3. Click the check box next to the message type to which you want to assign a transport, and click **Change**. If the message type is not in the list, click **New**. The Message Transport Assignment page opens.
4. If this is a new transport assignment, select the message type to which a transport is to be assigned from the **Message Type** drop-down list.
5. Type the transport configuration values in the appropriate fields. In general, a **message severity** of 0,0, and a Standard Device Format is recommended.
6. Click **Next** to configure the transport parameters for the specified message type.
7. Type the attributes for the transport you have chosen for this message type.
8. Click **Finish** or **Cancel** to return to the Message Type Configuration page.

Configuring a transport method for a site

To configure a transport method for your site, do the following:

1. Open the Administration Console and log on as a Site Administrator.
2. From the **Configuration** menu, select **Transports**. The Transport Configuration page displays.
3. Select the check box beside the method you want to configure.
4. Click **Configure**. The Transport Configuration Parameters dialog opens. The name of the transport method you selected appears at the top-left of the parameter table.
5. Type the values to be used by the transport method.
6. Click **OK** to accept the changes, or click **Cancel** to return to the Transport Configuration page.

Deactivating a transport method for a site

To deactivate an existing transport method for your site, do the following:

1. Open the Administration Console and log on as a Site Administrator.
2. From the **Configuration** menu, select **Transports**. The Transport Configuration page displays.
3. Click the check box next to the transport you want to deactivate.
4. Click **Change Status**. The page reloads, and the status changes.

Checking the system settings for the e-mail transport method

Ensure that the default settings for the e-mail transport method have been set by doing the following:

1. Launch the Configuration Manager.
2. Select the **Instance**, then open the **transports** —> **Outbound**—> **JavaMail** —> **ConnectionSpec** folder.
3. Click the **Advanced** tab in the right-hand frame.
4. Set the value of **host**, to your SMTP mail server.
5. Set the value of **protocol**, to `smtp`.
6. Click **Apply**.

7. Restart the WebSphere Commerce Server.

Note: This provides a system wide default setting for the e-mail transport. This setting will be overridden by those created in the Administration Console.

Activating notification

Once you have set the system default settings for the e-mail transport method, activate notification by doing the following:

1. Edit the <*instance_name*>.xml by setting:

```
<Notification display="false">
<order enabled="true" />
<error enabled="true" />
<pwdreset enabled="true" />
</Notification>
```

2. Restart WebSphere Commerce Server.

Note: If you encounter problems when sending e-mail for error notification the following actions may be required:

- Check that configuration values are correct.
- Clean-up all configuration data by deleting all entries that configure transport at the store level. The following SQL statement will delete all entries:

```
delete from cseditatt where store_id is not null
```

- Open the Administration Console and check that all message types are configured properly and do not configure transport at the store level.

Enabling error notification

To enable e-mail error notification, do the following:

1. Ensure that the system default settings for the e-mail transport method have been set.
2. Activate notification for error messages.
3. Assign the error condition message type to a transport.
4. Assign a transport method to message types for a site.

   On the Message Transport Assignment page enter the following values:

   a. On the **Message Type** drop-down list select Description of an Error Condition occurring in WebSphere Commerce.
   b. On the **Message Severity** field specify 0 0.
   c. From the **Transport** drop-down list, select E-mail.
   d. From the **Device Format** drop-down list, select Standard Device Format.

   On the Message Transport Assignment parameter page use the following values:

   a. On the **Host** field, type the fully qualified name of your mail server.
   b. On the **Protocol** field, type smtp.
   c. On the **Recipient** field, specify the administrator who should receive error notification messages. If multiple recipients are specified, separate recipient names with commas.
   d. On the **Sender** field, specify the sender of the message.
      This text appears in the Sender field of the e-mail message.
   e. On the **Subject** field, specify the subject of the message.
      This text appears in the Subject field of the e-mail message.

Note: If you encounter problems when sending e-mail for error notification the following actions may be required:

- Check that configuration values are correct.
- Clean-up all configuration data by deleting all entries that configure transport at the store level. The following SQL statement will delete all entries:

```
delete from cseditatt where store_id is not null
```

- Open the Administration Console and check that all message types are configured properly and do not configure transport at the store level.

Activating notification

Once you have set the system default settings for the e-mail transport method, activate notification by doing the following:

1. Edit the <*instance_name*>.xml by setting:

```
<Notification display="false">
<order enabled="true" />
<error enabled="true" />
<pwdreset enabled="true" />
</Notification>
```

2. Restart WebSphere Commerce Server.

Note: If you encounter problems when sending e-mail for error notification the following actions may be required:

- Check that configuration values are correct.
- Clean-up all configuration data by deleting all entries that configure transport at the store level. The following SQL statement will delete all entries:

```
delete from cseditatt where store_id is not null
```

- Open the Administration Console and check that all message types are configured properly and do not configure transport at the store level.

Enabling the shipment notification e-mail

To enable the shipment notification email to be sent to customers when their order is shipped, do the following:

- Update the CMDREG database table using the following SQL statement:

```
update cmdreg set classname='com.ibm.commerce.messaging.commands.ReleaseShipNotifyCmdImpl'
where interfacename='com.ibm.commerce.messaging.commands.ReleaseShipNotifyCmd'
```

Note: By default, the Release_WCS_ShipmentNotify message is disabled using ReleaseShipNotifyDummyImpl as the CLASSNAME in the CMDREG database table. The ReleaseShipNotifyDummyImpl is a dummy implementation which does nothing.

Enabling broadcast messages

To send a broadcast e-mail message, do the following:

1. Create a JSP file called BroadcastMessage.jsp.
2. Place the file in the site or store directory.
3. Open the Administration Console and log on as a Site or Store Administrator.

a. From the **Configuration** menu, select **Message Types**. The Message Type Configuration page displays.
b. Click **New**. The Message Transport Assignment page opens.
   1) On the **Message Type** drop-down list select A broadcast message.
   2) On the **Message Severity** field specify 0 0.
   3) From the **Transport** drop-down list, select E-mail.
   4) From the **Device Format** drop-down list, select Standard Device Format.
c. Click **Next**. The Message Transport Assignment parameter page displays.
   1) On the **Host** field, type the fully qualified name of your mail server.
   2) On the **Protocol** field, type `smtp`.
   3) On the **Recipient** field, specify a default recipient. If multiple recipients are specified, separate recipient names with commas. The recipient name will be replaced by the customers e-mail address at run-time.
   4) On the **Sender** field, specify the sender of the message. This text appears as the content of the From line in the e-mail.
   5) On the **Subject** field, specify the subject of the message. This text appears as the content of the Subject line in the e-mail.
d. Click **Finish**.
e. To send the message, follow the directions in the BroadcastMessage command reference file. For example, to send a message to all customers who have purchased part number “sku1234” from any store in site type the following on the address line of your browser:

```
BroadcastMessage?subject=testing&messageContent=this+is+a+test
    &sender=example%40ca.ibm.com&mode=2&partNumber=sku1234&URL=BroadcastMessage.jsp
```

Enabling order status notification

To enable notification for order status messages, do the following:

1. Open a database command window on your WebSphere Commerce machine.
2. Register the order status notification command by typing the following SQL statement:
   ```
   update cmdreg set
   classname='com.ibm.commerce.messaging.commands.OrderStatusNotifySendCmdImpl'whereinterfacename='com.ibm.co
   ```
3. Stop and the WebSphere Commerce Server.
4. If necessary, add transports methods to your site or store.
5. Assign the “Notification message of the order status” type to a transport. Ensure that you select **HTTP Browser** in the **Device Format** drop-down list.

Setting up outbound message composition

To set up and use the composition service for an outbound message, do the following:

1. Assign the transports to the appropriate message type, using either of these methods:
   - Assigning a transport method to a message type for a site
   - Assigning a transport method to a message type for a store
     A valid device format, as specified in the DEVICEFMT table, must be specified for each transport to be used.
2. Referring to the information in the topic Outbound messaging system interface, create a messaging system object using the SendMsgCmd task command. Use the setMsgType() and setStoreId() initialization services.
3. Invoke the messaging system compose method.

Assigning the error condition message type to a transport

Ensure that error messages are assigned to the site, not individual stores. To assign the error condition message type to a transport method, do the following:

1. Open the Administration Console and log on as a Site Administrator.
2. From the **Configuration** menu, select **Message Types**. The Message Type Configuration page displays.
3. Click **New**. The Message Transport Assignment page opens.
   a. On the **Message Type** drop-down list select Description of an Error Condition occurring in WebSphere Commerce.
   b. On the **Message Severity** field specify 0 0.
   c. From the **Transport** drop-down list, select E-mail.
   d. From the **Device Format** drop-down list, select Standard Device Format.
4. Click **Next**. The Message Transport Assignment parameter page displays.
   a. On the **Host** field, type the fully qualified name of your mail server.
   b. On the **Protocol** field, type `smtp`.
   c. On the **Recipient** field, specify the administrator who should receive error notification messages. If multiple recipients are specified, separate recipient names with commas.
   d. On the **Sender** field, specify the sender of the message.
   This text appears in the Sender field of the e-mail message.
   e. On the **Subject** field, specify the subject of the message.
   This text appears in the Subject field of the e-mail message.
5. Click **Finish**.

Enabling order status notification

To enable notification for order status messages, do the following:

1. Open a database command window on your WebSphere Commerce machine.
2. Register the order status notification command by typing the following SQL statement:
   ```
   update cmdreg set
   classname='com.ibm.commerce.messaging.commands.OrderStatusNotifySendCmdImpl'whereinterfacename='com.ibm.comm
   ```
3. Stop and the WebSphere Commerce Server.
4. If necessary, add transports methods to your site or store.
5. Assign the "Notification message of the order status" type to a transport. Ensure that you select **HTTP Browser** in the **Device Format** drop-down list.

Enabling messages to be sent from the Administration Console

If you have based your store on the InFashion sample store, you can enable Customer Service Representatives to send messages to customers using the WebSphere Commerce Administration Console. To enable messages to be sent from the WebSphere Commerce Administration Console, do the following:

1. Open the Administration Console and log on as a Site Administrator or Store Administrator.
2. From the **Configuration** menu, select **Message Types**. The Message Type Configuration page displays.
3. Click **New**. The Message Transport Assignment page opens.
   a. On the **Message Type** drop-down list select Order related message sent by customer service representative.
   b. On the **Message Severity** field specify 0 0.

c. From the **Transport** drop-down list, select E-mail.
d. From the **Device Format** drop-down list, select Standard Device Format.

4. Click **Next**. The Message Transport Assignment parameter page displays.
   a. On the **Host** field, type the fully qualified name of your mail server.
   b. On the **Protocol** field, type `smtp`.
   c. On the **Recipient** field, specify a default recipient. If multiple recipients are specified, separate recipient names with commas. The recipient name will be replaced by the customer e-mail address at run-time.
   d. On the **Sender** field, specify the sender of the message.
   This text appears in the Sender field of the e-mail message. (This value is overriden by the E-mail address value entered in Store Services.)
   e. On the **Subject** field, specify the subject of the message.
   This text appears in the Subject field of the e-mail message. (This value is overriden by the values entered in Store Services.)
5. Click **Finish**.

Enabling the outbound OrderCreate message

The OrderCreate outbound message can be enabled in either XML format or the legacy format. The two formats are generated by the following task commands:

1. `SendXMLOrder` generates the Report_NC_PurchaseOrder XML message.
2. `SendWCSOrder` generates the Order Create legacy message.

It is recommended that you use the XML format unless you are migrating from a previous version of WebSphere Commerce and want to maintain the existing format. You cannot enable both.

To enable the Report_NC_PurchaseOrder XML message, update your database CMDREG table using the following SQL statement:

```
update cmdreg set classname='com.ibm.commerce.messaging.commands.SendXMLOrderCmdImpl' where
interfacename='com.ibm.commerce.order.commands.OrderMessagingCmd'
```

To enable the Order Create legacy message, update your database CMDREG table using the following SQL statement:

```
update cmdreg set classname='com.ibm.commerce.messaging.commands.SendWCSOrderCmdImpl' where
interfacename='com.ibm.commerce.order.commands.OrderMessagingCmd'
```

Chapter 7. Message types

The WebSphere Commerce outbound messaging system can process different message types. Each message type is sent to the messaging system in response to a specific type of event that occurs within the WebSphere Commerce system. The messaging system processes the message according to the message type and the message settings specified in the Administration Console. The following table shows the message types supported by the outbound messaging system.

Message Type in MSGTYPE table	**Name in Administration Console**	**Usage**
ErrorMessage	Description of an error condition occurring in WebSphere WebSphere Commerce	Configure this message type to enable administrators to receive e-mail messages when an error occurs in WebSphere Commerce. To enable this message, see Enable error notification.
OrderCreateFixFormat	Outbound message for WebSphere Commerce order create	Indicates that an order has been created in WebSphere Commerce. The message can be used to send an outbound WebSphere Commerce order create message to a back-end system. See Enable the outbound OrderCreate message.
OrderCreateXMLFormat	Outbound message for WebSphere Commerce XML create	Indicates that an order has been created in WebSphere Commerce. The message can be used to send an outbound WebSphere Commerce order create message to a back-end system. See Enable the outbound OrderCreate message.
OrderStatusNotify	Notification message of the order status	Indicates that the status of an order has changed. See Enable order status notification.
OrderAuthorized	Message for an authorized order	Indicates that an order has been authorized. To send authorized order messages you need to create a JSP template. See OrderProcess command.
OrderReceived	Message for a received order	Indicates that an order has been received. To send received order messages you will need to create a JSP template. See OrderProcess command.

Message Type in MSGTYPE table	Name in Administration Console	Usage
OrderRejected	Message for a rejected order	Indicates that an order has been rejected. To send rejected order messages you need to create a JSP template. See OrderProcess command.
OrderCancel	Notification message for a canceled order	Indicates that an order has been canceled. To send canceled order messages you will need to create a JSP template. See OrderProcess command.
PasswordNotify	Notification message for password reset	Configure this message type to enable e-mail messages to be sent to customers indicating that their password has been reset.
BroadcastMessage	A broadcast message	Configure this message type to send broadcast message to customers. To send broadcast messages you need to create a broadcast message JSP template. See BroadcastMessage command.
MerchantOrderNotify	Message for notifying the merchant of an order	Related to the NotifyMerchant parameter of the OrderProcess command. To send notification messages you need to create a JSP template. See OrderProcess command.
AdminOrderComment	Order related message sent by customer service representative	Configure this message type to enable Customer Service Representatives to send customers e-mail messages from the WebSphere Commerce Accelerator.
NotifyReleaseShip	Message sent to customer to notify them their order has been shipped	Configure this message type to send customers e-mail messages about shipment confirmation of orders. See the ReleaseShipConfirm command.

Outbound messaging system interface

Interactions with the outbound messaging system can be done through the SendMsgCmd task command. This task command externalizes all available interfaces to methods provided by the messaging system to set necessary parameters to construct and use the messaging system object. The following is a list of all available services in the SendMsgCmd command interface:

- Initialization services
- Message content setting services

- Add e-mail parts or attachments services
- Configurable message data services
- Sending services
- Other services

Outbound message extension

The Report_NC_PurchaserOrder message includes the UserData XML element, which contains the UserDataField element. You can customize the UserData element to pass extra information not included in the Report_NC_PurchaserOrder outbound message by extending the SendXMLOrderMsg command and implementing either the getHeaderExtensionRecords() method or the getItemExtensionRecords()method, depending on whether you want to add the information at the header or item level. By default, both getHeaderExtensionRecords() and getItemExtensionRecords() methods return a null String value. Once implemented, both methods should return a String object that contains a series of UserDataField elements as follows:

```
<UserDataField name="field_name">field_value</UserDataField>
    .
    UserDataField repeated loop
    .
<UserDataField name="field_name">field_value</UserDataField>
```

For more information about how to extend commands, see the *WebSphere Commerce Programmer's Guide*.The following table outlines the methods called for the XML elements of the Report_NC_PurchaseOrder message:

Message	XML Element	Method Called
Report_NC_PurchaserOrder	Report_PO_Header	getHeaderExtensionRecords()
Report_NC_PurchaserOrder	Report_PO_Item	getItemExtensionRecords()

The Order Create message includes the USRLST record, which contains the DATUSR_DATA data segment. DATUSR allows you to add optional fields to a message. You can customize the DATUSR records to pass extra information not included in the Order Create outbound message by extending the SendWCSOrderMsg command and implementing either the getHeaderExtensionRecords() method or the getItemExtensionRecords()method, depending on whether you want to add the information at the header or item level. By default, both getHeaderExtensionRecords() and getItemExtensionRecords() methods return a null String value. Once implemented, both methods should return a String object that contains a series of DATUSR records as follows:

For more information about how to extend commands, see the *WebSphere Commerce Programmer's Guide*. The following table outlines the methods called for the sections of the Order Create message:

Message	USRLST Section	Method Called
Order Create	<HEADER>	getHeaderExtensionRecords()
Order Create	<ITMDAT>	getItemExtensionRecords()

New outbound message support

In addition to the supported XML and WebSphere Commerce outbound messages, you can add support for new outbound messages. To add a new outbound message, you must write a new controller command to build the content of the new outbound message and send the message to the back-end system using the send services of the outbound messaging system.

There are two methods for building the content of new outbound messages. The first method is to build your new outbound message in your own String buffer and assign it to the outbound messaging system through the use of its message content setting services that set the message content directly. This alternative requires you to include the logic of building the message in the controller command that you write. The second method is to build your new outbound message through the use of the outbound messaging system composition services. Through the use of JSP templates, the composition services generate an outbound message according to the message layout and content you defined in the JSP template.

For more information on how to use the outbound messaging services, refer to the Outbound Messaging System Interface section. For more information on how to write commands, see the *WebSphere Commerce Programmer's Guide*.

UserData element for outbound messages

The outbound XML message Report_NC_PurchaseOrder includes the UserData XML element as an optional element. Include the UserDataField element in this message to send additional data. You can customize the fields to pass extra data that is not included in the messages.

The name of the new field to be added should be the attribute of the name for the UserDataField element.

The following DTD describes the UserData element:

```
<!ELEMENT UserData (UserDataField+)>
<!ELEMENT UserDataField (#PCDATA)>
<!ATTLIST UserDataField
name CDATA #REQUIRED>
```

The following is an example of the UserData element:

```
<UserData>
   <UserDataField name="field_name">field_value</UserDataField>
            .
            .UserDataField repeated loop
            .
</UserData>
```

Outbound messaging system interface programming examples

The following Java code segment shows how interactions with the outbound messaging system can take place. Example 1 shows you how to build a new XML message and send it through the outbound messaging system. Example 2 shows you how to build an e-mail message and send it through the outbound messaging system:

Example 1

```
try
{
```

```
com.ibm.commerce.messaging.commands.SendMsgCmd api =
(com.ibm.commerce.messaging.commands.SendMsgCmd)
CommandFactory.createCommand(SendMsgCmd.NAME, getStoreId());
// Assume you have set the msgType in the MSGTYPES table to 100 and you are using
// storeId of 1.
api.setMsgType(new Integer(100));
api.setStoreID(new Integer(1));

// You have to choice on how to build the msg:
// First choice: build your xml msg in a String object and then use the setContent().
String OrderCreateMsg = new String("<?xml version="1.0" encoding="UTF-8"?> ...");
api.setContent(OrderCreateMsg);

// Or, use the message composition services (compose()) by passing the template/view name
// This view name should be registered in VIEWREG and MSGTYPES tables referring to
// a JSP message layout template.
String viewName = new String("OrderCreateMsgView");
TypedProperty tp = new TypedProperty();
// get the orderRefNumber and put it into tp
tp.put("ORDER_REF_NUMBER", getOrderRn().toString());
// get the languageId and put it into tp
tp.put("LANGUAGE_ID", getCommandContext().getLanguageId());
// Pass the viewName, command Context and parameters stored in tp to compose services.
// Upon successful completion, a message is build according to message layout defined in the
// JSP message layout template referred by viewName.
api.compose(viewName, getCommandContext(), tp);

// Send out the message using sendTransacted send service.
api.sendTransacted();
// Set the command context obtained from the controller command.
api.setCommandContext(getCommandContext());
// Run the outbound messaging system services
api.execute();
}
catch (Exception ex )
{
ex.printStackTrace(System.err);
}
```

Example 2

```
try
{
com.ibm.commerce.messaging.commands.SendMsgCmd api =
(com.ibm.commerce.messaging.commands.SendMsgCmd)
CommandFactory.createCommand(SendMsgCmd.NAME, getStoreId());
// Assume you have set the msgType in the MSGTYPES table to 200 and you are using
// storeId of 1.
api.setMsgType(new Integer(200));
api.setStoreID(new Integer(1));

// You have to choice on how to build the msg:
// First choice: build your xml msg in a String object and then use the setContent().
String OrderNotifyMsg =
new String("Your Order has been received. Thank You for Shopping with us.");
api.setContent(OrderNotifyMsg);
```

```
// Or, use the message composition services (compose()) by passing the template/view name
// This view name should be registered in VIEWREG and MSGTYPES tables referring to
// a JSP message layout template.
String viewName = new String("OrderNotifyMsgView");
TypedProperty tp = null;
// Pass the viewName, command Context and null parameter stored in tp to compose services.
// Upon successful completion, a message is build according to message layout defined in the
// JSP message layout template referred by viewName.
api.compose(viewName, getCommandContext(), tp);

// Set the subject, recipient and sender information using Configurable message data services
api.setConfigData("subject","Your Order has been received");
api.setConfigData("recipient",getEmailAddress());
api.setConfigData("sender","storeAdmin@storeABC.com);
// Send out the message using sendImmediate send service.
api.sendImmediate();
// Set the command context obtained from the controller command.
api.setCommandContext(getCommandContext());
// Run the outbound messaging system services
api.execute();
}
catch (Exception ex )
{
ex.printStackTrace(System.err);
```

Message composition templates

The WebSphere Commerce outbound messaging system includes JavaServer page composition templates for a number of message types. When a message of one of these types is generated within WebSphere Commerce, the message composition service uses the corresponding template to create the outbound message. Once it is created, the outbound message can be sent through whatever transports have been assigned to the message type in the Administration Console. An example of a message type that uses a message composition template is OrderCreateXMLFormat, which uses the `OrderCreateXML.jsp` template. The JavaServer page templates can be found in the following directory:

2000

drive:\Program Files\WebSphere\CommerceServer\installedApps\WC_Enterprise_App<*instance_name*>.ear\wcstores.war

NT

drive:\WebSphere\CommerceServer\installedApps\WC_Enterprise_App<*instance_name*>.ear\wcstores.war

AIX

/usr/WebSphere/CommerceServer/installedApps/WC_Enterprise_App<*instance_name*>.ear/wcstores.war

Solaris

/opt/WebSphere/CommerceServer/installedApps/WC_Enterprise_App<*instance_name*>.ear/wcstores.war

400

/QIBM/Userdata/WebASAdv4/<WAS_*instance_name*>installedApps/WC_Enterprise_App<*instance_name*>.ear/wcstores.war

The following table shows message types that use composition templates, and the JSP file associated with it.

Message type	Template
OrderCreateXMLFormat	OrderCreateXML.jsp
OrderStatusNotify	OrderStatusNotify.jsp
PasswordReset	PasswordResetNotification.jsp

You can customize these JSP files.

The following table shows other message types that use composition templates, to use these message types create your own JSP templates with the default name (the default name is registered in the VIEWREG table):

Message type	Default template name
OrderAuthorized	OrderAuthorized.jsp
OrderReceived	OrderReceived.jsp
OrderRejected	OrderRejected.jsp
BroadcastMessage	BroadcastMessage.jsp
MerchantOrderNotify	MerchantOrderNotification.jsp
OrderCancel	OrderCanceledNotification.jsp

Initialization services

These methods set the initial parameters that identify which messaging profile is used for the current message. These parameters retrieve the information created and maintained in the Administration Console.

- `public void setMsgType(Integer msgType)`
  This method is required. It is used to set the Message Type for the current message.
- `public void setStoreID(Integer storeId)`
  This method is required. It is used to retrieve message profile information for the store. To retrieve site-level information, the site-level store id can be used. The messaging system attempts to retrieve a profile based on the store entered. If none exists, it attempts to retrieve a profile based on the default site identifier.
- `public void setPriority(Integer priority)`
  This method provides optional initialization information. The priority integer specified limits the profiles retrieved. Only those profiles whose priority range includes this integer will be retrieved for the current message.

Message content setting services

You can either use the composition service or set the message content directly. To use the messaging system composition service, use the following service:

- To use the messaging system composition service, use the following service:
  - `public void compose( String viewName, CommandContext cmdContext, TypedProperty inParms)`
    This method accesses the composer functionality. It allows users to set the message content through the use of JSP templates. See the composer documentation for more information on this topic. The cmdContext parameter provides the necessary context information to the composer. The viewName parameter allows the user to determine the JSP that will be accessed by the composer. The inParms parameter represents the data to be passed to the JSP. There are rules governing what vales can be placed in the TypedProperty object. See the composer documentation for more information.
- To set the message content directly, use the following services:

- `public void setContent(Integer transportId, Integer languageId, byte[] msgContent)`
  This method allows you to set the content of a message directly using the msgContent parameter. The other two parameters must be present, but can be null. The transportID parameter allows you to set the content for a specific transport. The languageID parameter allows you to set the content for a specific language. For maximum flexibility in setting the content for multiple languages, it is suggested that you use the compose method to run a JSP.
- `public void setContent(Integer transportId, Integer languageId, String msgContent)`
  This setContent method performs the same function as the one described above, except it allows you to enter the content in String format rather than as an array of bytes.

Add e-mail parts or attachments services

The standard e-mail transport, as well as some other transports, allow attachments to be added to messages. The following methods allow users of the messaging system to attach content parts, or attachments, to messages:

- `public void addContentPart(byte[] msgAttachment)`
  This method offers a simple way to add content parts to a message. For the e-mail transport, a “content part” refers to an attachment. The msgAttachment parameter represents the content to be added to the message. Note that this must be the actual content of the part, translated into byte format.
- `public void addContentPart(byte[] msgAttachment, String partName, String partType)`
  This method gives you greater flexibility in adding content parts to a message. The msgAttachment parameter represents the content to be added to the message. The partName parameter represents a name to be used for the content part. The partType parameter represents the MIME type of the part being sent. For example, the partType for Mime Email could be ’text/plain’.

Configurable message data services

Use the following generic method to configure the transport services used for the message:

- `public void setConfigData(String key, String value)`
  This generic method allows the user to configure the transport services used for the message. The key parameter refers to the administration name used to identify the attribute to be changed. The value parameter is the value to be assigned. Invoking this method to this method will cause the values specified here to override the values assigned in the Administration Console. Refer to the table below containing the default transports available to the messaging system, and the attributes that apply to each. See the addMember method below for an alternative way of setting e-mail recipients.

The default transports available to the Messaging System contain the following attributes (attribute keys are case sensitive):

Transport	Attribute key	Description
E-mail	subject	The subject of the email.
	recipient	The e-mail address of the recipient.
	sender	The e-mail address of the sender.
	host	The mail host used to send the message.
	protocol	The protocol used to connect to the mail host.
File	location	The location of the file to be written.
	FileName	The name of the file to be written.
	mode	The type of write to perform. 0 - append, or create if the file does not exist 1 - overwrite

Sending services

The following methods are provided with the outbound messaging system sending services:

- `public void sendImmediate()`
  This method sends the message immediately to recipients. The caller is blocked until the message has been sent.
- `public void sendTransacted()`
  This method stores the message in the MSGSTORE database table. At a predetermined time, the WebSphere Commerce scheduler invokes a job that sends all messages stored in batch mode. Using this method ensures that a send occurs only after the caller has committed or terminated successfully. This method should be used if blocking a call using the sendImmediate() method cannot be tolerated.
- `sendReceiveImmediate()`
  This method is used to perform a request-reply send. This type of send is used with the MQ-JMS transport for back-end integration messages. The content of the reply is stored internally and can be accessed via the getReply() method.
  Hint: To perform a send-receive using the MQ-JMS transport, you must ensure that you have set the mode attribute appropriately, using either the Administration Console or the setConfigData() method in the configurable message data services.
- `public byte[] getReply()`
  This method is used to retrieve the result of the sendReceiveImmediate() method. To obtain the result, it should be called after the performExecute() method, which executes the command. It returns the response from the transport as an array of bytes.

Other services

The following methods outline the other services offered by the outbound messaging system:

- `public void addMember(Long aMember)`
  This method represents the second way of entering recipient data into the messaging system (the first being setConfigData). The parameter represents a valid member. The user can call this method repeatedly, each call adding an additional member to the list of recipients. Internally, the messaging system extracts the appropriate address from the Member.
  Note 1: It is important that you ensure validation is performed on member addresses.
  Note 2: The recipient can be set using either addMember or setConfigData but not both. If both are entered, the setConfigData entries will be overridden by the addMember entries. Also note that the messaging system requires the user to enter the recipients before the content is set. This is because member information may alter the content used in a message.
- `public void remTransport(Integer aRemoveTransportID)`
  This method gives you flexibility over which transports are used to deliver the method. The parameter should be a transport id that is valid for this message type. When this method is called, the messaging system removes the transport from the messages list of transports that may be used. In other words, the transport passed in as a parameter will be disabled for this particular message call.
- `public void setPartialSend(Boolean partialSend)`
  This method is applicable only when the addMember method is used to set the recipients. The boolean parameter represents whether partial sends are permitted. In the messaging system, a partial send refers to sending the message as long as one of the members added has a preferred e-mail address associated with their profile. The following is an explanation of the partialSend parameter:
  - **true**: Permit partial sends. Those members who have not set up an address in the appropriate place will be skipped.
  - **false**: All or nothing. Returns an exception if even one of the members does not have an appropriately configured email.
- **Note**: No verification is performed by the messaging system to ensure an address is of the correct format. Partial send operates on the principle of the existence of a value in the appropriate place.

Chapter 8. Inbound back-end integration messages

An inbound message is a request that WebSphere Commerce receives from an external application. Each inbound message activates a command in WebSphere Commerce that performs a particular function. If there is an error processing an inbound message, it is placed into the error queue.

WebSphere Commerce supports inbound back-end messages that accomplish the following five functions:

- Create a customer registration
- Update a customer registration
- Update the status of an order
- Update the inventory for a product
- Update the price of a product

Each of the functions listed above can be activated by a request message in XML format, and some can be activated using the legacy message format. In general, the XML format is recommended. The XML messages are encoded in UTF-8 format.

To create a customer registration, use the Create_WCS_Customer XML message. If you already record customer information on an existing back-end system, rather than re-create this information from scratch, use this message to register the customer data on the WebSphere Commerce database. The message sends existing customer information from the back-end server to the WebSphere Commerce server. You can also use the Create_NC_Customer XML message, and the Customer Update legacy messages to perform similar function. However, the Create_WCS_Customer XML message provides you with capability to enter more customer information.

To update a customer registration that already exists in the WebSphere Commerce database, use the Update_WCS_Customer XML message. When you use this message, the back-end customer management system updates customer information and sends the message to WebSphere Commerce to update information about a registered shopper. You can also use the Update_NC_Customer XML message, and the Customer Update legacy messages to perform a similar function. However, the Update_WCS_Customer XML message provides you with capability to update more customer information.

To update the status of an order that already exists in the WebSphere Commerce database, use the Update_WCS_OrderStatus XML message. Use this message to update the WebSphere Commerce database with the status of orders that are processed by a back-end application. For example, the shipping status of an order from the back-end fulfillment system can be updated in the WebSphere Commerce system by sending this message from the back-end system to WebSphere Commerce. You can also use the Update_NC_OrderStatus XML message or the Order Status Update legacy message to perform similar function. However, the Update_WCS_OrderStatus XML message provides you with capability to update more order status information.

To update the inventory for a product that already exists in the WebSphere Commerce database, use the Update_WCS_ProductInventory XML message. This message is ideal for instances when an external or back-end inventory system maintains product inventory. The external system can send the message to WebSphere Commerce to update inventory in the WebSphere Commerce database. You can also use the Product Quantity Update legacy message to perform the same function. Alternatively, the Update_NC_ProductInventory XML message performs a similar function. However, the Update_WCS_ProductInventory XML message provides you with capability to update more inventory information.

To update either the list price or the offer price of a product that already exists in the WebSphere Commerce database, use the Update_WCS_ProductPrice XML message. This message is ideal for instances when an external or back-end system maintains product information, including prices. The

external system can send one of these messages to WebSphere Commerce to update list prices or offer prices of products in the WebSphere Commerce database. If you are updating the offer price of a product, you can also use the Update_NC_ProductPrice XML message or the Product Price Update legacy message to perform the same function. However, only the Update_WCS_ProductPrice message can be used to update list price information.

Inbound fulfillment integration messages

An inbound fulfillment integration message is a request that WebSphere Commerce receives from a fulfillment center system. Each inbound message activates a command in WebSphere Commerce that performs a particular function. If there is an error processing an inbound message, the failed message is placed into the error queue.

WebSphere Commerce supports fulfillment integration messages that accomplish the following functions:

- Create an expected inventory record
- Create a pick batch
- Request pick tickets and packing lists
- Create a receipt
- Adjust the inventory receipt level of a product maintained by WebSphere Commerce database under the RECEIPT table
- Issue shipment confirmation by the fulfillment center

Each of the functions listed above can be activated by a message in XML format. The XML messages are encoded in UTF-8 format.

The following table outlines the inbound fulfillment integration messages used and the controller command that they invoke:

XML Messages	Description	Controller Command
Create_WCS_ExpectedInventoryRecord	Creates an expected inventory record.	ExpectedInventoryRecordCreateCmd
Create_WCS_PickBatch	Generates a pick batch.	PickBatchGenerateCmd
Inquire_WCS_PickPackListDetail	Requests details created by CreatePickBatch.	GetPickPackListDetailCmd
Create_WCS_InventoryReceipt	Creates inventory records of items.	ReceiptCreateCmd
Update_WCS_InventoryReceipt	Adjusts inventory for an item.	InventoryAdjustCmd
Create_WCS_ShipmentConfirmation	Issues shipment confirmation for an item.	ReleaseShipConfirmCmd

Creating an expected inventory record

To create an expected inventory record in the WebSphere Commerce database, use the Create_WCS_ExpectedInventoryRecord XML message. This message is ideal for instances when a fulfillment center manages the inventory and ordering information of vendors. WebSphere Commerce is informed about the availability of future stock and can track inventory levels. The fulfillment center can send one of these messages to WebSphere Commerce to create an expected inventory record in the WebSphere Commerce database when the inventory level for a product is low. This record can be used for backorders.

Creating a pick batch

To create a pick batch record in the WebSphere Commerce database, use the Create_WCS_PickBatch XML message. This message is ideal for instances when a fulfillment center manages the pickbatch functions. The fulfillment center can send one of these messages to WebSphere Commerce to create a

pickbatch in the WebSphere Commerce database. A pickbatch groups together all outstanding orders ready for release for the given fulfillment center and store. A pick ticket for all released orders in a pickbatch is created and stored in the PICKBATCH table. A packing list for each released order is created and stored in ORDRELEASE table.

Getting pick pack list details
To inquire about the pick ticket details, use the Inquire_WCS_PickPackListDetail XML message. This message is used with the Create_WCS_PickBatch message. After the fulfillment center sends a Create_WCS_PickBatch message to WebSphere Commerce, a pickbatch is created, the new pickbatch ID is returned to the fulfillment center. The fulfillment center can then send the Inquire_WCS_PickPackListDetail message with the pick batch ID as a parameter to retrieve the details of the pick ticket and packing list.

Creating inventory receipts
To create a receipt for products ordered, use the Create_WCS_InventoryReceipt XML message. This message allows you to create a receipt for products that have been ordered from a vendor, helping to update the inventory available on hand (under the RECEIPT database table) within the WebSphere WebSphere Commerce inventory database. A fulfillment center can send one of these messages to WebSphere Commerce to create a receipt that can be used to keep track of products ordered.

Updating inventory levels
To resolve any discrepancies between a physical inventory count and the inventory levels maintained in WebSphere Commerce, use the Update_WCS_InventoryReceipt XML message. This message is used when an external fulfillment center system manages the inventory shipments. The fulfillment center system can send this message to WebSphere Commerce to adjust product inventory levels.

Issuing shipment confirmation
To issue a shipment confirmation to WebSphere Commerce, use the Create_WCS_ShipmentConfirmation XML message. A fulfillment center can send one of these messages to WebSphere Commerce to create a shipment confirmation message verifying that an order is shipped. This message also gives you the option of sending an e-mail notification to the customer when orders are shipped. See Enabling the Shipment Notification e-mail for further details on how to enable the customer e-mail notification.

Chapter 9. Adding a new inbound XML message

The following steps are required to add support for a new inbound message:

1. Define a DTD for the new XML message. You can use the DTD files for existing XML messages as a guide. By default these files are located in the following directory:

   2000

   drive:\Program Files\WebSphere\CommerceServer\xml\messaging

   NT

   drive:\WebSphere\CommerceServer\xml\messaging

   AIX

   /usr/WebSphere/CommerceServer/xml/messaging

   Solaris

   /opt/WebSphere/CommerceServer/xml/messaging

   400

   /QIBM/ProdData/WebCommerce/xml/messaging
2. Add the new DTD file to the system.
3. Update the `user_template.xml` inbound message template definition file for the new message. To do this, refer to the structural guidelines outlined in Inbound message template definition files.

Adding a new DTD file to the system

To allow your new inbound XML message to be recognized and processed by the XML message mapper, do the following:

1. Place the DTD file you created for the new message in the same directory as your other DTD files. By default, the directory is:

   2000

   drive:\Program Files\WebSphere\CommerceServer\xml\messaging

   NT

   drive:\WebSphere\CommerceServer\xml\messaging

   AIX

   /usr/WebSphere/CommerceServer/xml/messaging

   Solaris

   /opt/WebSphere/CommerceServer/xml/messaging

   400

   /QIBM/ProdData/WebCommerce/xml/messaging
2. Add the name of the new DTD file to the list of inbound message DTD files.

Adding to the list of inbound message DTD files

To add a new DTD file to the list of DTD files for inbound messages, do the following:

1. Launch the Configuration Manager.
2. Select **Instance Properties**, then open the **Messaging** folder.

3. In the Inbound Message DTD Files field add the name of your new DTD file to the end of the list, placing a comma before the new filename.
4. Click **Apply** to save the changes.
5. From the WebSphere Application Server Administration Console, stop then re-start the instance.

Inbound message extension

WebSphere Commerce allows you to modify or extend the functionality of all inbound messages by modifying the WebSphere Commerce controller command that is run by each message. You can provide additional pre-processing or post-processing statements to any inbound message command used, or you can override the existing processing entirely. To do this, you need to have a knowledge of Java programming.

When an inbound message is received from a back-end system, its information is processed into command parameters and a WebSphere Commerce controller command is invoked along with all the provided parameters. When the command is run, the performExecute() method is invoked, which in turn invokes three methods, in the following order:

1. doPreProcess()
2. doProcess()
3. doPostProcess()

When you first install WebSphere Commerce, only the doProcess() method contains programming statements. You can add pre-processing statements by extending the command and implementing the doPreProcess() method, or you can add post-processing statements by implementing the doPostProcess() method. Alternatively, you can implement either the doProcess() or the performExecute() method to overwrite the entire process. For more information about how to extend commands, see the *WebSphere Commerce Programmer's Guide*.

UserData element for inbound messages

All inbound messages include the UserData XML element as an optional element.. Include the UserDataField element to transport additional data. You can customize the fields to pass extra data that is not included in the messages.

The name of the new field to be added should be the attribute of the name for the UserDataField element.

The following DTD describes the UserData element:

```
<!ELEMENT UserData (UserDataField+)>
<!ELEMENT UserDataField (#PCDATA)>
<!ATTLIST UserDataField
name CDATA #REQUIRED
>
```

The following is an example of the UserData element:

```
<UserData>
   <UserDataField name="field_name">field_value</UserDataField>
            .
            .UserDataField repeated loop
            .
</UserData>
```

Chapter 10. Message mappers

A message mapper is a mechanism that takes an XML message and converts it to a CommandProperty object. It provides a common interface so that messages can be converted to CommandProperty objects and used by all WebSphere Commerce components.

Supported adapters, such as the Program Adapter, and WebSphere Commerce components can both call a message mapper. For both, the message mapper performs the following tasks:

- Receives an XML message.
- Converts the message to a CommandProperty object.
- Returns null if the the XML message cannot be converted.

Inbound messages are sent to WebSphere Commerce by back-end systems or external systems to request some sort of action. In order for WebSphere Commerce to perform that action, the XML message must be processed by the message mapper to determine what action has been requested. WebSphere Commerce includes two message mappers at installation: the XML message mapper for parsing XML integration messages, and the legacy message mapper for parsing back-end integration legacy messages.

The CommandProperty object represents a WebSphere Commerce command to accommodate the requirements from the supported device adapters. Other components can also use the message mapper mechanism to convert messages to CommandProperty objects.

The lifecycle of a message mapper exists throughout the WebSphere Commerce instance. It is initialized when an instance is started and it resides as long as the instance runs.

XML message mapper

The XML message mapper is responsible for converting the XML data from inbound XML messages to CommandProperty objects. It is an extension of the ECSAX parser. The XML configuration node for the XML message mapper found in the *instance_name*.xml configuration file should look similar to the following:

```
<MessageMapper messageMapperId="-1"
classname="com.ibm.commerce.messaging.programadapter.messagemapper.ecsax.ECSAXMessageMapper"
    enable="true"
    name="WCS.INTEGRATION">
<configuration/>
</MessageMapper>
```

Legacy message mapper

The legacy message mapper is responsible for converting the data from inbound legacy messages to CommandProperty objects. The XML configuration node for the legacy message mapper found in the *instance_name*.xml configuration file should look similar to the following:

```
<MessageMapper messageMapperId="-2"
classname="com.ibm.commerce.messaging.programadapter.messagemapper.nclegacy.NetCMessageMapper"
    enable="true">
    name="NC.LEGACY">
<configuration>
</MessageMapper>
```

Inbound message template definition files

WebSphere Commerce provides an XML message mapper which can be used to map inbound XML messages to WebSphere Commerce command interfaces based on the inbound XML message template definition files.

Whenever an inbound XML message is passed to the message mapper, it checks to see if the message is defined in the template definition files. If so, it retrieves the WebSphere Commerce controller command name and parameter names for the message, and parses the incoming message to get the values for the parameters. Once the message has been parsed, the message mapper returns a CommandProperty object that contains the command name and the parameter name-value pairs for the command.

The message template definition files are used to define the XML parsing information for the inbound XML message. Each message defined in these files has the following two base elements:

- **TemplateDocument**: Defines the DTD file used by the message, the command that is called when the message is received, which tag mapping is to be used, and the XML element from which tag mapping is started.
- **TemplateTag**: Defines the mapping of XML elements in the DTD file to parameter names of commands in WebSphere Commerce. The template tag element identifies the parameter names and tells the message mapper where to find the values on an incoming message.

There are two template definition files provided by WebSphere Commerce. The `sys_template.xml` file is the template definition used to map existing WebSphere Commerce inbound XML messages. The `user_template.xml` is provided to enable you to add additional inbound XML messages. Both files are in XML format, based on the `ec_template.dtd` template definition DTD file.

NT

drive:\WebSphere\CommerceServer\xml\messaging

2000

drive:\Program Files\WebSphere\CommerceServer\xml\messaging

AIX

/usr/WebSphere/CommerceServer/xml/messaging

Solaris

/opt/WebSphere/CommerceServer/xml/messaging

400

/QIBM/Proddata/WebCommerce/xml/messaging

Important: For security reasons, you must ensure that only authorized persons can access and modify the `sys_template.xml` and the `user_template.xml` message template definition files. If unauthorized persons have access to write to this file, they would have the ability to write new inbound messages that could invoke any WebSphere Commerce command as a Site Administrator.

Removing message mappers

To remove a message mapper, you must manually remove it from the group of message mappers in the *instance_name*.xml configuration file. To remove a message mapper, do the following:

1. Open the *instance_name*.xml configuration file.
2. Locate the component with the name MessageMapperGroup.
3. Locate the XML configuration node for the message mapper that you want to remove. It will look similar to the following:

```
<MessageMapper messageMapperId="#"
        classname="class inplementing MessageMapper interface"
        enable="true"
        name="Name of Message Mapper">
    <configuration />
    </MessageMapper>
```

4. Change the enable parameter to "false". This will disable the message mapper and make it unavailable for use.

Adding message mappers

To add a new message mapper, you must manually add it to the group of message mappers in the *instance_name*.xml configuration file. To add a new message mapper, do the following:

1. Open the *instance_name*.xml configuration file.
2. Locate the component with the name MessageMapperGroup.
3. Between the property tags of that component, add the following XML node to define your message mapper:

```
<MessageMapper messageMapperId="#"
        classname="class inplementing MessageMapper interface"
        enable="true"
        name="Name of Message Mapper">
</MessageMapper>
```

   Refer to Message mapper configuration for information on these parameters.

4. Within the configuration node of the message mapper, add any extra configuration parameters needed for the message mapper. This is converted into a TypedProperty object and passes to the init method of the message mapper. The following is an example of extra parameters that might be added:

```
<configuration
EcSystemTemplateFile="mapping.xml"
EcInboundMessageDtdFiles="something.dtd"
EcTemplatePath="E:\users\user\test\map"
EcSaxParserClass="com.ibm.xml.parsers.ValidatingSAXParser"
EcInboundMessageDtdPath="E:\users\user\test\dtd"
EcSaxParserClass="com.ibm.xml.parsers.ValidatingSAXParser"
/>
```

New inbound message support

In addition to the supported XML and WebSphere Commerce messages, you can add support for new inbound messages. There are two primary methods for adding new inbound messages.

The recommended method is to add a new inbound XML message through the use of the user_template.xml inbound message template definition file. In this file, you can indicate the controller command the new inbound message invokes, define the elements of the message, and indicate the command parameters to which each element corresponds. When the message is received, the XML

message mapper identifies the command to be run and the parameters to be used. The command is then invoked using the Site Administrator authority. For security reasons, you must ensure that only authorized persons can access and modify the `user_template.xml` message template definition file, otherwise unauthorized users would have the ability to write a new inbound message and invoke any WebSphere Commerce command as Site Administrator.

If you do not want to use the inbound XML message template definition files together with the XML message mapper, you can also implement the NewInboundMessage command to add new messages. This command is invoked when the message mapper does not recognize the message as an existing legacy message, or as an XML message defined in the inbound XML message template definition files. Since the NewInboundMessage command is not pre-programmed, you have full control over the processing that takes place once it is invoked. However, this method requires considerable programming effort, particularly where there are a large number of new messages.

Chapter 11. Customizing the NewInboundMessage command

To customize the NewInboundMessage command to process messages that you have created, do the following:

1. Extend the NewInboundMessage command. Refer to the *WebSphere Commerce Programmer's Guide* for details on how to do this.
2. To receive the inbound message, use the getMessage() method of the command, which returns the message as a String.
3. Implement the performExecute() method of the command. Inside the method, place the programming statements that process your inbound message.
4. Use the following SQL statement to register your new extended command by updating the CMDREG table in the WebSphere Commerce database:

```
update cmdreg set classname='yourCommandClassName' where
interfacename='com.ibm.commerce.messaging.commands.NewInboundMessageCmd'
```

Chapter 12. Message mapper configuration

The *instance_name*.xml configuration file lists all possible message mappers, and includes their name, class, device format, whether they are available, and specific configuration parameters. To locate the parameters for a certain message mapper, find the component of the *instance_name*.xml file with the name MessageMapperGroup.Since each message mapper may have different configuration parameters, each has a node of XML configuration information within the MessageMapperGroup component. The only requirement for the format of the configuration parameters node is that the parameters must be name-value-pairs. This allows the message mapper configuration parameters to be easily converted to a TypeProperty object.

The following lists the parameters for each individual message mapper:

- name: the name of the message mapper.
- class: the class that contains the implementation of the message mapper interface.
- messageMapperId: the ID for the message mapper. Each message mapper ID must be unique.
- enabled: indicate whether the message mapper should be used and initialized or not. If this value is set to "true", the message mapper is initialized at the startup of the instance and is available. If this value is set to "false", the message mapper is disabled.

XML parsing using template definition files

When the XML message mapper parses an inbound XML message, it gets the document type, the version if it is available, and the element name, one by one from the XML documents in the message. The message mapper looks up the template document defined in the ECTemplate element of the template definition file. The template document tells the message mapper the following information:

- From which element the tag mapping should be started.
- Which tag template to use.
- The command name to be invoked by the inbound message. The command could be an existing WebSphere Commerce controller command or a new one that you have created.

Once the tag mapping has started, the message mapper looks in the TemplateTag, as defined in the ECTemplate file to determine the field name and type based on the XPath generated from the inbound XML message, then sets a value for that field. The field and value pairs are stored in a TypedProperty hash table of either commandProperty or messageProperty, based on the value in the FieldInfo attribute for that field. After the whole inbound message is parsed successfully, a PropertyCommand object is returned, which contains the command name as well as the commandProperty and messageProperty objects.

sys_template.xml file

The sys_template.xml file contains the outline of all inbound XML messages supported by WebSphere Commerce. The file defines the data fields for each message, mapping the message to the appropriate WebSphere Commerce Controller Command, and mapping each field within the message to the appropriate parameter for that command. The structure of sys_template.xml is based on the ec_template.dtd file, which defines the format that messages must take.

Do not add new messages to this file. To add your own inbound messages, use the user_template.dtd file.

All XML files are located in the following directory:

2000

drive:\Program Files\WebSphere\CommerceServer\xml\messaging

NT

drive:\WebSphere\CommerceServer\xml\messaging

AIX

/usr/WebSphere/CommerceServer/xml/messaging

Solaris

/opt/WebSphere/CommerceServer/xml/messaging

400

/QIBM/Proddata/WebCommerce/xml/messaging

user_template.xml file

The user_template.xml is an XML message template definition file that allows you to add new inbound XML messages to be supported by your system. An outline should be added to this file for each new XML message that you want to support. You can use the sys_template.xml file as a guide on how to use or update this template file.

The outline should indicate the tag template to be used, the element from which the tag mapping should be started, the name of the WebSphere Commerce controller command to be invoked, and the URL parameters that correspond to each XML element. If you are using the new inbound XML message to invoke a new WebSphere Commerce command, refer to the*WebSphere Commerce Programmer's Guide* on how to write and register a new WebSphere Commerce controller command.

All XML files are located in the following directory:

2000

drive:\Program Files\WebSphere\CommerceServer\xml\messaging

NT

drive:\WebSphere\CommerceServer\xml\messaging

AIX

/usr/WebSphere/CommerceServer/xml/messaging

Solaris

/opt/WebSphere/CommerceServer/xml/messaging

400

/QIBM/Proddata/WebCommerce/xml/messaging

Before you add any of your own messages, ensure the file contains the following lines:

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

Each message that you define in the template definition file includes two base elements:

- TemplateDocument
- TemplateTag

For an explanation of each of these elements, follow the link below.

ec_template.dtd file

The ec_template.dtd file contains the blueprint for all inbound messages outlined in the `sys_template.xml` file, and should be used to guide you in the creation of any new messages in the `user_template.xml` file.

All DTD files are located in the following directory:

2000

drive:\Program Files\WebSphere\CommerceServer\xml\messaging

NT

drive:\WebSphere\CommerceServer\xml\messaging

AIX

/usr/WebSphere/CommerceServer/xml/messaging

Solaris

/opt/WebSphere/CommerceServer/xml/messaging

400

/QIBM/Proddata/WebCommerce/xml/messaging

The following text describes the `ec_template.dtd` file:

```
<!ELEMENT ECTemplate ((TemplateDocument | TemplateTag)*)>
<!ATTLIST ECTemplate
version CDATA #FIXED "1.0">
```

TemplateDocument element of a template definition file

For each inbound XML message, there must be at least one template document defined in the message. This is defined in the TemplateDocument element. The TemplateDocument element has four sub-elements:

- **DocumentType (with optional version attribute):** This part specifies the XML document name, or root element name, and the “version” attribute of the root element, if it exists. The following is an example of the DocumentType element:
  `<DocumentType version='1.0'>Reset_Password</DocumentType>`
- **StartElement:** This part specifies the element from which the XML message mapper starts the tag mapping. This must correspond to an element in the DTD for the message. The tag mapping generates the name value pairs which are used as command parameters. The following is an example of the

usage of StartElement:
```
<StartElement>PasswordInfo</StartElement>
```
In this example, there should be an element named `PasswordInfo` defined in the DTD file for this message. For example, the DTD file should contain a line such as the following that defines a PasswordInfo element:
```
<!ELEMENT PasswordInfo (Password, ConfirmPassword)>
```

- **TemplateTagName:** This part specifies which tag mapping is to be used for the message. The tag mapping is defined elsewhere in the template definition file under the `TemplateTag` element. This means that the value of the `TemplateTagName` element should match the `name` attribute of a `TemplateTag` element that appears somewhere in the template definition file. Refer to the section on the `TemplateTag` element for more details. The following example illustrates the use of the TemplateTagName element:
  ```
  <TemplateTagName>PasswordReset10Map</TemplateTagName>
  ```
  In this example, the following TemplateTag element should exist elsewhere in the template definition file:
  ```
  <TemplateTag name='PasswordReset10Map'>
  ...
  </TemplateTag>
  ```
  The elements between the start and end tags depend on the mapping of the message.
- **CommandMapping:** This element determines which commands are invoked by the message. The element contains one sub-element called `Command`. The `Command` element is used to indicate the WebSphere Commerce controller command that is run when the message is successfully mapped. It contains one mandatory attribute called `CommandName`, which is used to indicate the name of the command. The value of this attribute must correspond to an existing command that is registered in the URLREG table. For example, you could associate a message with the command that resets a password using the following syntax:
  ```
  <CommandMapping>
    <Command CommandName='ResetPassword'/>
  </CommandMapping>
  ```
  You can also associate multiple commands with the same message. To do this, you need to use the following additional attributes of the Command element:
  - **Condition attribute**: The syntax for condition is as follows ([] means it is optional, * means it can be repeated):

    ```
    fieldName ["fieldValue "] [AND fieldName ["fieldValue "]]*
    ```

    fieldName

    It should match the `Field` attribute of the `Tag` element in the TemplateTag definition. If the `XPath` attribute of this tag exists in the inbound XML message, then the condition is true.

    fieldName="fieldValue "

    A value in the XML message is set to the field in `fieldName`. When the message is mapped, if the value is same as `fieldValue`, the condition is true.

    fieldName1="fieldValue1" AND fieldName1="fieldValue2"

    The values in the XML message are set to the field `fieldName1` and `fieldName2` when the message is parsed. If the values are same as `fieldValue1` and `fieldValue2` respectively, then the condition is true.
  - **TemplateTagName attribute**: If you specify the TemplateTagName attribute for this Command element, whenever the Condition becomes true, a new tag template with the name defined in TemplateTagName will be used for the rest of the inbound XML message.
  - **Constant Element**: the list of constants to be put in the TypedProperty for that command.
    - **Field**: The field name of the name value pair which will be put into the TypedProperty.
    - **FieldInfo**: See the definition in TemplateTag.
      The value should be placed between the `<Constant>` and `</Constant>` tags.

The following is a simple example of how you could use the multiple Command elements to map to multiple commands:

```
<CommandMapping>
  <Command CommandName='ResetPassword' Condition='Verb="Reset" AND Noun="Password"'/>
  <Command CommandName='AdminResetPassword' Condition='Verb="Reset" AND Noun="AdminPassword"'/>
</CommandMapping>
```

In this example, you there should be Noun and Verb elements defined in the DTD file for the message.

TemplateTag element of a template definition file

The `TemplateTag` element is used, along with the `TemplateDocument` element, in the template definition file. Several `TemplateTag` elements can be defined for each inbound XML message. However, each `TemplateTag` element must be linked to a `TemplateDocument` element using the `name` attribute. The value associated with the `name` attribute should match the value in the `TemplateTagName` sub-element of a `TemplateDocument` element defined with the template definition file.

Each `TemplateTag` element contains a list of tag definitions in the `Tag` sub-element. The `Tag` element can contain the following five attributes:

- **XPath**: The path of the XML element, relative to the `StartElement` indicated in the `TemplateDocument`. The `XPath` element is the key to finding the field name of the name-value pair. Examples of XPath are:
  - E1/E2: Element E2, which is nested inside element E1. The following is an example of this type of path:
    `XPath='Address/ZipCode'`
    E1/E2/E3: Element E3, which is nested inside element E2, which is nested inside element E1. The following is an example of this type of path:
    `XPath='ContactInfo/Address/ZipCode'`
  - E3@a1 : The attribute a1 in element E3. The following is an example of this type of path:
    `XPath='InvoiceInfo@InvoiceType'`
    In this case, the `InvoiceType` attribute may be used to indicate which one out of a variety of invoice types the message applies to.
  - E4[1] : The first instance of element E4. There can be multiple instances of E4. For example, you could use this format where you have multiple lines of an address. Each line of the address would be contained in a separate Tag element.
  - E5[@a2="value 2"] : An instance of element E5, where the attribute a2 of E5 equals to "value 2". For example, you could use the following format:
    `XPath='InvoiceInfo[@InvoiceType="ShippingInvoice"]`
    E5[@a1="value1"][@a2="value 2"] : The instance of element E5, where the attribute a1 of E5 equals to '"value1", and the attribute a2 of E5 equals to "value 2".
- **XPathType**: The type of element indicated in the XPath. This attribute indicates how the XML element should be processed by the XML parser. The supported types are:
  - **PCDATA** : The element or the attribute of the element contains raw, inbound data that will be processed and returned in a nave-value pair. This is the default value for the XPathType.
  - **EMPTY**: The element is empty or contains data that can be ignored. No name-value pair is returned for this element.
  - **REPEAT** : The element can have multiple instances. Each element's PCDATA is returned in a name-value pair.
  - **ATTRIBUTE**: If the element field name is determined by the value of an attribute, this attribute should have ATTRIBUTE type.
  - **VECTOR**: A new hash table will be appended to the Vector, and all name-value pairs generated for sub-elements will be put into the new hash table.
  - **USERDATA**: Indicates a user-defined element. The element has an attribute called `name` whose value is the field name. The data of the element is returned in a name-value pair.

- **Field**: The field name of the name-value pair which will be put into the `TypedProperty`. This should match name of a parameter used by the called command.
- **FieldType**: The type of the data field. The field type could be `String` or `Date` (ISO 8601 Date format). `String` is default.
- **FieldInfo**: Indicate the `TypedProperty` into which the name value pair should be placed. `Data` is the default. If you want to put the name value pair into more than one `TypedProperty`, you must specify more than one of the values listed below, separated by a comma:
  - **Data**: The name-value pair will be put into the a `commandProperty` object which contains arguments for the command.
  - **Control**: The name-value pair will be put into a `messageProperty` which contains control information for the command, such as USERID or PASSWORD
  - **Command**: The name-value pair is used to determine which command should be called. The generated name-value pairs are used in the `CommandMapping` element of the `TemplateDocument` element.

If the element XPath is not found in the tag template, the XPath in the XML message will be used as the field name, and the XPathType is PCDATA, generating a name-value pair using XPath as the field name.

For an example of how the `TemplateTag` element is used, refer to the `sys_template.xml` file.

Chapter 13. Messaging system back-end integration messages

The WebSphere Commerce messaging system provides a mechanism for integrating WebSphere Commerce with back-end systems through the use of inbound and outbound messages. Inbound messages are used to run commands in WebSphere Commerce based on messages coming from back-end systems. Outbound messages can be generated by the outbound messaging system in order to update back-end systems with events that have taken place, such as a new customer order. To use back-end integration messages, you must have an adapter installed, and have the messaging system configured to receive XML messages.

The messaging system is prepared to send and receive a number of pre-defined messages in XML format. This format offers a high degree of readability, making the messages easy to modify and maintain. You can also use the legacy message format. However, the XML message format is recommended. For an explanation of each message, refer to the sections on inbound and outbound back-end integration messages. You can also add new messages. For new inbound messages, you can associate them with either existing WebSphere Commerce commands, or commands that you have created.

Chapter 14. Fulfillment integration messages

WebSphere Commerce provides a mechanism for integration with fulfillment center systems using inbound and outbound messages. Inbound fulfillment integration messages are used to run commands in WebSphere Commerce based on inbound requests received from fulfillment center systems. Outbound messages can be generated by the outbound messaging system in order to update fulfillment center systems with events that have taken place, such as receipt of new stock, or an order shipment. To use fulfillment integration messages, you must have an adapter installed, and have the messaging system configured to receive XML messages.

The messaging system is prepared to send and receive a number of pre-defined messages in XML format. This format offers a high degree of readability, making the messages easy to modify and maintain. For an explanation of each message, refer to the sections on inbound and outbound fulfillment integration messages. You can also add new messages. For new inbound messages, you can associate them with either existing WebSphere Commerce commands, or commands that you have created.

The format of the XML messages consists of a set of XML elements defined within specific DTD files. Each DTD may contain one or more common files, identified by a `.mod` file extension. In addition, each inbound message is associated with a WebSphere Commerce controller command in the `sys_template.xml` message template definition file. All DTD, MOD, and XML files are located in the following directory:

2000

drive:\Program Files\WebSphere\CommerceServer\xml\messaging

NT

drive:\WebSphere\CommerceServer\xml\messaging

AIX

/usr/WebSphere/CommerceServer/xml/messaging

Solaris

/opt/WebSphere/CommerceServer/xml/messaging

400

QIBM/ProdData/WebCommerce/xml/messaging

Chapter 15. Customizing the NewInboundMessage command

To customize the NewInboundMessage command to process messages that you have created, do the following:

1. Extend the NewInboundMessage command. Refer to the *WebSphere Commerce Programmer's Guide* for details on how to do this.
2. To receive the inbound message, use the getMessage() method of the command, which returns the message as a String.
3. Implement the performExecute() method of the command. Inside the method, place the programming statements that process your inbound message.
4. Use the following SQL statement to register your new extended command by updating the CMDREG table in the WebSphere Commerce database:

```
update cmdreg set classname='yourCommandClassName' where
interfacename='com.ibm.commerce.messaging.commands.NewInboundMessageCmd'
```

Chapter 16. Integration message DTD files

All supported WebSphere Commerce integration XML messages consist of information found in DTD files. Some DTD files use information from the common file `NCCommon.mod` or other MOD files. The format and the source of the XML element values for the DTD files are described in these MOD files.

All DTD and MOD files are located in the following directory:

2000

drive:\Program Files\WebSphere\CommerceServer\xml\messaging

NT

drive:\WebSphere\CommerceServer\xml\messaging

AIX

/usr/WebSphere/CommerceServer/xml/messaging

Solaris

/opt/WebSphere/CommerceServer/xml/messaging

400

/QIBM/Proddata/WebCommerce/xml/messaging

Message	DTD and MOD Files used
Create_NC_Customer message	Create_NC_Customer_10.dtd NCCommon.mod NCCustomer_10.mod
Update_NC_OrderStatus message	Update_NC_OrderStatus_10.dtd NCCommon.mod
Update_NC_ProductInventory message	Update_NC_ProductInventory_10.dtd NCCommon.mod
Update_NC_ProductPrice message	Update_NC_ProductPrice_10.dtd NCCommon.mod
Report_NC_PurchaseOrder message	Report_NC_PO_10.dtd
Create_WCS_Customer message	Create_WCS_Customer_20.dtd NCCommon.mod
Update_WCS_ProductPrice message	Update_WCS_ProductPrice_20.dtd NCCommon.mod
Update_WCS_ProductInventory message	Update_WCS_ProductInventory_20.dtd NCCommon.mod.
Update_WCS_Customer message	Update_WCS_Customer_10.dtd NCCommon.mod NCCustomer_10.mod
Update_WCS_OrderStatus message	Update_WCS_OrderStatus_20.dtd NCCommon.mod
Create_WCS_ExpectedInventoryCreate message	Create_WCS_ExpectedInventoryRecord_10.dtd

Create_WCS_PickBatch message	Create_WCS_PickBatch_10.dtd
Inquire_WCS_PickPackListDetail message	Inquire_WCS_PickPackListDetail_10.dtd
Create_WCS_InventoryReceipt message	Create_WCS_InventoryReceipt_10.dtd
Update_WCS_InventoryReceipt message	Update_WCS_InventoryReceipt_10.dtd
Create_WCS_ShipmentConfirmation message	Create_WCS_ShipmentConfirmation_10.dtd
Response_WCS_ExpectedInventoryRecord message	Response_WCS_ExpectedInventoryRecord_10.dtd
Response_WCS_PickBatch message	Response_WCS_PickBatch_10.dtd
Report_WCS_PickPackListDetail message	Report_WCS_PickPackListDetail_10.dtd
Response_WCS_CreateInvReceipt message	Response_WCS_CreateInvReceipt_10.dtd
Response_WCS_UpdateInvReceipt message	Response_WCS_UpdateInvReceipt_10.dtd
Response_WCS_CreateShipConfirm message	Response_WCS_CreateShipConfirm_10.dtd

Back-end integration legacy messages

The WebSphere Commerce offers support for messages which use the legacy message format. Unless you are migrating from a previous version of WebSphere Commerce Suite, it is recommended that you use the XML messages instead, since they accomplish the same function, and are easier to read and maintain.

The format of the WebSphere Commerce messages consists of two sections: the message descriptor and the application data. In the case of inbound messages, the message descriptor contains control information required to operate, such as the message identity and type. The application data contains the information to be processed. All WebSphere Commerce messages consist of a set of tags and records in a logical sequence and defined data segments within the records. The <PROLOG>, <HDR>, <ITM>, and <DATUSR> records, which are included in the supported messages, adhere to the following format:

```
<TAG>DATA SEGMENT</TAG>
```

where the data segment is identified with a _DATA suffix. For instance, a record for the Order Create message looks like this:

```
<HDR010>ORDER_CREATE_HDR010_DATA</HDR010>
```

Each data segment (in this example, ORDER_CREATE_HDR010_DATA) must be replaced with specific field and database table information for the particular message.

The following table outlines the six supported messages, as well as the controller command called by each one:

Message Name	Message Type	Data Segment	Controller Command
Customer New	Inbound	CUSTOMER_NEW_PROLOG_DATA CUSTOMER_NEW_HDR010_DATA DATUSR_DATA for inbound messages	UserRegistration Add
Customer Update	Inbound	CUSTOMER_UPDATE_PROLOG_DATA CUSTOMER_UPDATE_HDR010_DATA DATUSR_DATA for inbound messages	UserRegistration Update

Message Name	Message Type	Data Segment	Controller Command
Order Create	Outbound	ORDER_CREATE_PROLOG_DATA ORDER_CREATE_HDR010_DATA ORDER_CREATE_HDR020_DATA ORDER_CREATE_HDR030_DATA ORDER_CREATE_HDR040_DATA DATUSR_DATA for outbound messages ORDER_CREATE_ITM010_DATA	SendWCSOrder
Order Status Update	Inbound	ORDER_STATUS_UPDATE_PROLOG_DATA ORDER_STATUS_UPDATE_HDR010_DATA DATUSR_DATA for inbound messages ORDER_STATUS_UPDATE_ITM010_DATA	OrderStatus
Product Price Update	Inbound	PRODUCT_PRICE_UPDATE_PROLOG_DATA PRODUCT_PRICE_UPDATE_HDR010_DATA	ProductOffer PriceUpdate
Product Quantity Update	Inbound	PRODUCT_QUANTITY_UPDATE_PROLOG_DATA PRODUCT_QUANTITY_UPDATE_HDR010_DATA	ProductInventory Update

Back-end integration XML messages

WebSphere Commerce offers support for inbound and outbound messages that use the XML format. Each inbound message invokes specific behaviors within the WebSphere Commerce Server by executing a controller command. Each controller command in turn performs operations on the WebSphere Commerce database and subsystems. Some controller commands can be executed by more than one XML message. In addition, some messages can invoke different commands, depending on the content of the message.

The format of the XML messages consists of a set of XML elements defined within specific DTD files. Each DTD may contain one or more common files, identified by a `.mod` file extension. In addition, each inbound message is associated with a WebSphere Commerce controller command in the `sys_template.xml` message template definition file. All DTD, MOD, and XML files are located in the following directory:

2000

drive:\Program Files\WebSphere\CommerceServer\xml\messaging

NT

drive:\WebSphere\CommerceServer\xml\messaging

AIX

/usr/WebSphere/CommerceServer/xml/messaging

Solaris

/opt/WebSphere/CommerceServer/xml/messaging

400

/QIBM/Proddata/WebCommerce/xml/messaging

The following table outlines the inbound messages used and the controller command that they invoke:

XML Messages	Description	Controller Command
Create_WCS_Customer, Create_NC_Customer	Creates a registration record for a new user, or updates a record for an existing user.	UserRegistrationAdd
Update_NC_Customer, Update_WCS_Customer	Updates a registration record for an existing user.	UserRegistrationUpdate
Update_WCS_OrderStatus, Update_NC_OrderStatus	Updates the general status of an order.	OrderStatus
Update_WCS_OrderStatus	Updates the confirmation status of an order.	OrderConfirmStatus
Update_WCS_OrderStatus	Updates the shipping status of an order.	OrderShippingStatus
Update_WCS_OrderStatus	Updates the invoice status of an order.	OrderInvoiceStatus
Update_WCS_ProductPrice, Update_NC_ProductPrice	Updates the offer price information for a product.	ProductOfferPriceUpdate
Update_WCS_ProductPrice	Updates the list price information for an order.	ProductListPriceUpdate
Update_NC_ProductInventory Update_WCS_ProductInventory	Updates product inventory information.	ProductInventoryUpdate

Note: Some messages contain the letters NC in the name and others contain the name WCS in the name. Those with names that contain the letters NC are XML messages from previous versions of WebSphere Commerce. Those with WCS in the name use updated formats that offer greater flexibility. It is generally recommended that you use the WCS versions where you have choice.

The following table outlines the back-end integration message used by the outbound messaging system, as well as the command that generates it:

XML Message	Description	Controller Command
Report_NC_PurchaseOrder	Sends a message to a back-end system containing information on a new order.	SendXMLOrder

Sample scenarios using fulfillment integration messages

Fulfillment integration messages allow WebSphere Commerce to communicate with a fulfillment center system. This allows a Site Administrator to stay informed about the availability of products they are offering to customers. The following scenarios illustrate the way a fulfillment center system and the WebSphere Commerce system can work together by communicating using fulfillment integration messages.

Scenario 1 - Expected Inventory & Backorders

A fulfillment center system finds the inventory level of an item is low. It orders more inventory from a vendor and uses the Create_WCS_ExpectedInventoryRecord XML message to report the expected receipt of new stock to WebSphere Commerce.

WebSphere Commerce can continue to offer that item for sale, even if the inventory level is low, by allowing backorders based on the expected receipt of more stock. WebSphere Commerce sends a Response_WCS_ExpectedInvRecord message in response that includes a `WCSRaDetailID` parameter.

The fulfillment center notes this WCSRaDetailID parameter. When the new shipment arrives, the fulfillment center sends the Create_WCS_InventoryReceipt XML message including the WCSRaDetailID parameter for reference. If there is no WCSRaDetailID associated with the new stock, the WCSRaDetailID can be omitted.

Scenario 2 - Inventory Update

The staff at the fulfillment center discovers a discrepancy between an inventory level recorded in the system and the actual inventory present during a physical inventory count. The fulfillment center can use the Update_WCS_InventoryReceipt XML message to inform WebSphere Commerce of the discrepancy.

WebSphere Commerce updates the recorded inventory levels accordingly and responds with the Response_WCS_UpdateInvReceipt message.

Scenario 3 - Order Fulfillment

To fulfill an order, the fulfillment center sends the Create_WCS_PickBatch XML message to WebSphere Commerce, initiating the fulfillment process.

WebSphere Commerce sends a response message including a PickBatchID, grouping together a list of “ready to ship” items.

The fulfillment center sends the Inquire_WCS_PickPackListDetail XML message to request pick ticket and packing list details, as well as other shipping information.

WebSphere Commerce responds with a list of what to pick and pack.

The fulfillment center prepares the shipment and sends it to a customer, optionally informing WebSphere Commerce of the shipment with the Create_WCS_ShipmentConfirmation XML message. The fulfillment may also inform the customer of the shipment with an e-mail.

ReleaseShipNotify message

The ReleaseShipNotify message is an outbound e-mail message sent to notify the customer when an order release is manifested. This occurs when the STATUS column of the ORDRELEASE table is updated to MNF. The e-mail message is sent by the ReleaseShipNotify task command, using ReleaseShipNotify.jsp to compose the content of the message. The ReleaseShipNotify task command is invoked by the ReleaseManifest controller command.

This message can be used regardless of whether the fulfillment center system is internal or external. When using the internal WebSphere Commerce fulfillment center, an administrator can select the release manifest option in the shipment confirmation screen to trigger this message. When using an external fulfillment center system, a shipment confirmation message with the UpdateManifestStatus attribute set to 1 triggers this message.

The message can be enabled or disabled at the store level by overriding the ReleaseShipNotify task command. By default, this message is disabled using a ReleaseShipNotifyDummyImpl as the class name in the CMDREG table.

Response_WCS_ExpectedInvRecord message

The Response_WCS_ExpectedInvRecord message is an outbound message that contains information on a WebSphere Commerce expected inventory record. WebSphere Commerce generates this message in response to the inbound Create_WCS_ExpectedInventoryRecord message.

If the inbound message contains a valid StoreID or a valid ExpectedDate, the message calls the ExpectedInventoryRecordCreate command which redirects to either

ExpectedInventoryRecordCreateRedirectView view task on successful completion, or ExpectedInventoryRecordCreateErrorView view task on command failure. The ExpectedInventoryRecordCreateRedirectView view task is implemented by `RACreateError.jsp` to compose the Response_WCS_ExpectedInvRecord response message. Within the response message sent back to fulfillment centers, the `BackendRaDetailID` parameter can be included so that the fulfillment center can correctly associate the response with the original message it sent.

Note: If the inbound message does not contain a valid `StoreID`or a valid `ExpectedDate`, the GenericApplicationError viewname is used for error message composition. The response message is generated by `GenericApplicationErrorXML.jsp`.

The Response_WCS_ExpectedInvRecord message uses the XML message format and follows `Response_WCS_ExpectedInvRecord_10.dtd`.

The following table describes the format of the Response_WCS_ExpectedInvRecord message. For a description of the database column, follow the link to its associated table. All fields are optional unless otherwise noted. The tag value length in the XML message for database fields of type INT, BIGINT and DOUBLE should be CHAR (10), CHAR(19), and CHAR(16) respectively.

	Field Name	Comment	Table Name	Column Name	Note
1	ResponseStatus	Mandatory	N/A	N/A	Error comment in case of command failure
2	status	Mandatory	N/A	N/A	OK or ERROR (an attribute of ResponseStatus)
3	code		N/A	N/A	Error code (an attribute of ResponseStatus, existing only if status=“ERROR”)
4	BackendRaID		N/A	N/A	Referenced by the original Create_WCS_ExpectedInventoryRec message
5	StoreID		RA	STORE_ID	
6	VendorID		RA	VENDOR_ID	
7	OrderDate		RA	ORDERDATE	
8	WCSRaID		RA	RA_ID	
9	BackendRaDetailID		N/A	N/A	Can be used as a reference
10	ItemOwnerID		ITEMSPC	MEMBER_ID	
11	ProductSKU		ITEMSPC	PARTNUMBER	
12	WCSRaDetailID		RADETAIL	RADETAIL_ID	Can be used with Create_WCS_InventoryReceipt message

Response_WCS_PickBatch message

The Response_WCS_PickBatch message is an outbound message that contains information on a WebSphere Commerce pickbatch. WebSphere Commerce generates this message in response to the inbound Create_WCS_PickBatch message. If the inbound message contains a valid `StoreID`, the message calls the PickBatchGenerate command, which redirects to either PickBatchGenerateRedirectView view task on successful completion, or PickBatchGenerateErrorView view task on command failure. The PickBatchGenerateRedirectView is implemented by `PickBatchResult.jsp` for response processing. In the Response_WCS_PickPatch XML message, the back-end PickBatchID from the original request and the newly generated `PickBatchID` are sent back as the response. In case of command failure, PickBatchGenerateErrorView is used, which is implemented by `PickBatchError.jsp`.

Note: If the inbound message does not contain a valid `StoreID`, the GenericApplicationError viewname is used for error message composition. The response message is generated by `GenericApplicationErrorXML.jsp`.

The Response_WCS_PickBatch message uses the XML message format and follows `Response_WCS_PickBatch_10.dtd`.

The following table describes the format of the Response_WCS_PickBatch message. For a description of the database column, follow the link to its associated table. All fields are optional unless otherwise noted. The tag value length in the XML message for database fields of type INT, BIGINT and DOUBLE should be CHAR (10), CHAR(19), and CHAR(16) respectively.

Level	Field Name	Comment	Table Name	Column Name	Note
1	ResponseStatus	Mandatory	N/A	N/A	Error comment if status="ERROR"
2	status	Mandatory	N/A	N/A	OK or ERROR (an attribute of ResponseStatus)
3	code		N/A	N/A	Error code (an attribute of ResponseStatus, existing only if status="ERROR")
4	MorePickBatch		N/A	N/A	YES or NO: indicates whether another Create_WCS_PickBatch message should be submitted again for more PickBatches
5	BackendPickBatchID		N/A	N/A	A reference to the original PickBatch request
6	WCSPickBatchID		PICKBATCH	PICKBATCH_ID	Can be "NULL" if no PickBatch is available for the specified StoreID and FulfillmentCenterID. Used in the Inquire_WCS_PickPackListDetail message

Report_WCS_PickPackListDetail message

The Report_WCS_PickPackListDetail message is an outbound message that reports pick ticket and packing list details. WebSphere Commerce generates this message in response to the inbound Inquire_WCS_PickPackListDetail message. The inbound message calls the GetPickPackListDetail command which redirects to either PickPackListRedirectView view task on successful completion, or PickPackListErrorView view task on command failure. PickPackListRedirectView is implemented by PickPackListResult.jsp for response processing. PickPackListErrorView is implemented by PickPackListError.jsp.

The Report_WCS_PickPackListDetail message contains two individual fixed XML slips generated by the `Create_WCS_PickPackListDetail_10.dtd`. They are the pick ticket and the packing list. The attributes of pick tickets and packing lists are described in the tables below.

The Report_WCS_PickPackListDetail message uses the XML message format and follows `Report_WCS_PickPackListDetail_10.dtd`.

The following table describes the format of the Report_PickPackListDetail message. For a description of the database column, follow the link to its associated table. All fields are optional unless otherwise noted. The tag value length in the XML message for database fields of type INT, BIGINT and DOUBLE should be CHAR (10), CHAR(19), and CHAR(16) respectively.

Level	XML Element	Comment	Table Name	Column Name	Note
1	ResponseStatus	Mandatory	N/A	N/A	Error comment if `status="ERROR"`
2	status	Mandatory	N/A	N/A	OK or ERROR (an attribute of ResponseStatus)
3	code		N/A	N/A	Error code (an attribute of ResponseStatus, existing only if `status="ERROR"`)
4	PickPackListReport		N/A	N/A	Exists only if `status="OK"`. Contains pick ticket and packing lists.

The definition of the pick ticket element follows `PickTicket_10.dtd`. The content of this DTD file is fixed and should not be modified.

Level	XML Element	Comment	Table Name	Column Name	Note
1	Pickticket	Mandatory	N/A	N/A	Contains Pickbatch_Information and Pickbatch_Items. Multiple Pickbatch_Items may be found in 1 message.

Level	XML Element	Comment	Table Name	Column Name	Note
2	Pickbatch_Information	Mandatory	N/A	N/A	With attributes of Store_Name, Fulfillment_Center and Pickbatch_Number
3	Store_Name	Mandatory	STOREENTDS	DISPLAYNAME	An attribute of Pickbatch_Information
4	Fulfillment_Center	Mandatory	FFMCENTDS	DISPLAYNAME	An attribute of Pickbatch_Information
5	Pickbatch_Number	Mandatory	PICKBATCH	PICKBATCH_ID	An attribute of Pickbatch_Information
6	Pickbatch_Items		N/A	N/A	With attributes of SKU, Product_Name, Product_Description and Quantity
7	SKU	Mandatory	BASEITEM	PARTNUMBER	An attribute of Pickbatch_Items
8	Product_Name	Mandatory	BASEITMDSC	SHORTDESCRIPTION	An attribute of Pickbatch_Items
9	Product_Description		BASEITMDSC	LONGDESCRIPTION	An attribute of Pickbatch_Items
10	Quantity	Mandatory	ORDERITEMS	QUANTITY	An attribute of Pickbatch_Items

The definition of a packing list follows PackSlip_10.dtd. The content of this DTD file is fixed and should not be modified.

Level	XML Element	Comment	Table Name	Column Name	Note
1	Packslip	Mandatory	N/A	N/A	Contains Order_Information, Shipto and Order_Items. Multiple Packslip may be found in 1 message.
2	Order_Information	Mandatory	N/A	N/A	With attributes of Store_Name, Fulfillment_Center, Order_Number, Release_Number, PickBatch_Number, Order_Date, Catalog_Name, Shipping_Provider, Customer_Number and Invoice_Method
3	Store_Name	Mandatory	STOREENTDS	DISPLAYNAME	An attribute of Order_Information
4	Fulfillment_Center	Mandatory	FFMCENTDS	DISPLAYNAME	An attribute of Order_Information

Level	XML Element	Comment	Table Name	Column Name	Note
5	Order_Number	Mandatory	ORDERITEMS	ORDERS_ID	An attribute of Order_Information
6	Release_Number	Mandatory	ORDERITEMS	ORDERELEASENUM	An attribute of Order_Information
7	PickBatch_Number	Mandatory	PICKBATCH	PICKBATCH_ID	An attribute of Order_Information
8	Order_Date	Mandatory	ORDERS	TIMEPLACED	Attribute of Order_Information
9	Shipping_Provider	Mandatory	SHPMODEDSC	DESCRIPTION	An attribute of Order_Information
10	Customer_Number	Mandatory	ORDERS	MEMBER_ID	An attribute of Order_Information
11	Invoice_Method		TERMCOND	STRINGFIELD1	This is a sting. 3 options: both, e-Mail, printed. Default is NULL.
12	Shipto	Mandatory	N/A	N/A	With attributes of AddressID, First_Name, Last_Name, Middle_Name, Address_1, Address_2, Address_3, City, State, Zip and Country
13	AddressID	Mandatory	ADDRESS	ADDRESS_ID	An attribute of Shipto
14	First_Name		ADDRESS	FIRSTNAME	An attribute of Shipto
15	Last_Name	Mandatory	ADDRESS	LASTNAME	An attribute of Shipto
16	Middle_Name		ADDRESS	MIDDLENAME	An attribute of Shipto
17	Address_1	Mandatory	ADDRESS	MIDDLENAME	An attribute of Shipto
18	Address_2		ADDRESS	ADDRESS2	An attribute of Shipto
19	Address_3		ADDRESS	ADDRESS3	An attribute of Shipto
20	City	Mandatory	ADDRESS	CITY	An attribute of Shipto
21	State		ADDRESS	STATE	An attribute of Shipto
22	Zip		ADDRESS	ZIPCODE	An attribute of Shipto
23	Country	Mandatory	ADDRESS	COUNTRY	An attribute of Shipto

Level	XML Element	Comment	Table Name	Column Name	Note
24	Order_Items	Mandatory	N/A	N/A	May contain multiple components with attributes of SKU, Product_Name, Product_Description, Quantity, Catalog_Name, Unit_Price, Total_Price, Currency and Customer_Comments.
25	SKU	Mandatory	BASEITEM	PARTNUMBER	An attribute of Order_Items
26	Product_Name	Mandatory	BASEITMDSC	SHORTDESCRIPTION	An attribute of Order_Items
27	Product_Description		BASEITMDSC	LONGDESCRIPTION	An attribute of Order_Items
28	Quantity	Mandatory	ORDERITEMS	QUANTITY	An attribute of Order_Items
29	Catalog_Name	Mandatory	CATENTDESC	NAME	An attribute of Order_Items
30	Unit_Price	Mandatory	ORDERITEMS	PRICE	An attribute of Order_Items
31	Total_Price	Mandatory	ORDERITEMS	TOTALPRODUCT	An attribute of Order_Items
32	Currency	Mandatory	ORDERITEMS	CURRENCY	An attribute of Order_Items
33	Customer_Comments		ORDERITEMS	COMMENTS	An attribute of Order_Items
34	Component		N/A	N/A	With attributes of SKU, Product_Name, Product_Description and Quantity
35	SKU	Mandatory	BASEITEM	PARTNUMBER	An attribute of Component
36	Product_Name	Mandatory	BASEITMDSC	SHORTDESCRIPTION	An attribute of Component
37	Product_Description		BASEITMDSC	LONGDESCRIPTION	An attribute of Component
38	Quantity	Mandatory	ORDERITEMS	QUANTITY	An attribute of Component

Response_WCS_CreateInvReceipt message

The Response_WCS_CreateInvReceipt message is an outbound message that contains information for creating a WebSphere Commerce inventory receipt. WebSphere Commerce generates this message in response to the inbound Create_WCS_InventoryReceipt message. If the inbound message contains a valid `storeID`and a valid `ReceiptDate`, it calls the view task ReceiptCreateRedirectView which uses CreateInvReceiptOK.jsp to generate a response message. In the response message, a new receipt_id for

records created in the RECEIPT and RCPTAVAIL tables is included. If the command encounters an error, the view task used is ReceiptCreateErrorView. This error view task is implemented by CreateInvReceiptError.jsp. If enough inventory exists to fulfill an expected inventory record, the record is closed.

Note: If the inbound message does not contain a valid `StoreID`or a valid `ReceiptDate`, the GenericApplicationError viewname is used for error message composition. The response message is generated by `GenericApplicationErrorXML.jsp`.

The Response_WCS_CreateInvReceipt message uses the XML message format and follows `Response_WCS_CreateInvReceipt_10.dtd`.

The following table describes the format of the Response_WCS_CreateInvReceipt message. For a description of the database column, follow the link to its associated table. All fields are optional unless otherwise noted. The tag value length in the XML message for database fields of type INT, BIGINT and DOUBLE should be CHAR (10), CHAR(19), and CHAR(16) respectively.

Level	Field Name	Comment	Table Name	Column Name	Note
1	ResponseStatus	Mandatory	N/A	N/A	Error comment
2	status	Mandatory	N/A	N/A	OK or ERROR (an attribute of ResponseStatus)
3	code		N/A	N/A	Error code (an attribute of ResponseStatus, existing only if status="ERROR")
4	ItemOwnerID		ITEMSPC	MEMBER_ID	N/A
5	ProductSKU		ITEMSPC	PARTNUMBER	N/A
6	StoreID		RECEIPT	STORE_ID	N/A
7	FulfillmentCenterID		RECEIPT	FFMCENTER_ID	N/A
8	VendorID		RECEIPT	VENDOR_ID	N/A
9	QTYReceived		RECEIPT	QTYRECEIVED	N/A
10	ReceiptDate		RECEIPT	RECEIPTDATE	N/A

Response_WCS_UpdateInvReceipt message

The Response_WCS_UpdateInvReceipt message is an outbound message that contains information for updating the inventory for an item. WebSphere Commerce generates this message in response to the Update_WCS_InventoryReceipt message. If the inbound message contains a valid `StoreID`, it calls the InventoryAdjust command which redirects to the view task InventoryAdjustRedirectView. Upon successful completion, InventoryAdjustRedirectView uses `UpdateInvReceiptOK.jsp` to generate the response message. If the command encounters an error, the view task InventoryAdjustErrorView is used. This error view task is implemented by `UpdateInvReceiptError.jsp`.

If the inventory adjustment in the message is positive, the command creates a new row in the RECEIPT and RCPTAVAIL database tables. If the adjustment is negative, the QTYONHAND column in the RECEIPT table is marked down using the appropriate picking method.

Note: If the inbound message does not contain a valid `StoreID`, the GenericApplicationError viewname is used for error message composition. The response message is generated by `GenericApplicationErrorXML.jsp`.

The Response_WCS_UpdateInvReceipt message uses the XML message format and follows `Response_WCS_UpdateInvReceipt_10.dtd`.

The following table describes the format of the Response_WCS_UpdateInvReceipt message. For a description of the database column, follow the link to its associated table. All fields are optional unless otherwise noted. The tag value length in the XML message for database fields of type INT, BIGINT and DOUBLE should be CHAR (10), CHAR(19), and CHAR(16) respectively.

Level	Field Name	Comment	Table Name	Column Name	Note
1	ResponseStatus	Mandatory	N/A	N/A	Error comment
2	status	Mandatory	N/A	N/A	OK or ERROR (an attribute of ResponseStatus)
3	code		N/A	N/A	Error code (an attribute of ResponseStatus, existing only if status="ERROR")
4	ItemOwnerID		ITEMSPC	MEMBER_ID	N/A
5	ProductSKU		ITEMSPC	PARTNUMBER	N/A
6	StoreID		RECEIPT	STORE_ID	N/A
7	FulfillmentCenterID		RECEIPT	FFMCENTER_ID	N/A
8	QTYAdjusted		INVADJUST	QUANTITY	N/A
9	InvAdjCodeID		INVADJUST	INVADJCODE_ID	N/A

Response_WCS_CreateShipConfirm message

The Response_WCS_CreateShipConfirm message is an outbound message that contains information for creating a shipment confirmation for an order. WebSphere Commerce generates this message in response to the inbound Create_WCS_ShipmentConfirmation message. If the inbound message contains a valid `ActualShipDate`, it calls the ReleaseShipConfirm command, which redirects to the view task ReleaseShipConfirmRedirectView on successful completion. The Response_WCS_CreateShipConfirm response message is generated by `CreateShipConfirmOK.jsp`. The command updates the required database, changing the fulfillment status of the item to confirm shipment. It gets a new manifest_id from the MANIFEST table through the key manager, and propagates the MANIFEST table with input data. With Release_WCS_ShipmentNotify message enabled, if the command executes successfully and the UpdateManifestStatus is `1`, the default `ReleaseShipNotify.jsp` generates a notification email.

If the command encounters an error, it redirects to the view task ReleaseShipConfirmErrorView. This error view task is implemented by `CreateShipConfirmError.jsp`.

Note: If the inbound message does not contain a valid `ActualShipDate`, the GenericApplicationError viewname is used for error message composition. The response message is generated by `GenericApplicationErrorXML.jsp`.

The Response_WCS_CreateShipConfirm message uses the XML message format and follows `Response_WCS_CreateShipConfirm_10.dtd`.

The following table describes the format of the Response_WCS_CreateShipConfirm message. For a description of the database column, follow the link to its associated table. All fields are optional unless otherwise noted. The tag value length in the XML message for database fields of type INT, BIGINT and DOUBLE should be CHAR (10), CHAR(19), and CHAR(16) respectively.

Level	Field Name	Comment	Table Name	Column Name	Note
1	ResponseStatus	Mandatory	N/A	N/A	Error comment if status="ERROR"
2	status	Mandatory	N/A	N/A	OK or ERROR (an attribute of ResponseStatus)
3	code		N/A	N/A	Error code (an attribute of ResponseStatus, existing only if status="ERROR")
4	ShipModeID		MANIFEST	SHIPMODE_ID	N/A
5	OrderNumber		MANIFEST	ORDERS_ID	N/A
6	OrderReleaseNum		MANIFEST	ORDERRELEASENUM	N/A

Update_WCS_OrderStatus message

The Update_WCS_OrderStatus message is an inbound message that contains status information for a WebSphere Commerce order. The message has four possible forms. Although each form of the message has the same XML elements, each one is associated with a different Command. The mapping of message forms to commands is as follows:

Order Status Message Type	Command
OrderConfirm	OrderConfirmStatus
OrderShipping	OrderShippingStatus
OrderInvoice	OrderInvoiceStatus
OrderStatus	OrderStatus

When an order is received by a back-end system, it generates this message, containing any order fulfillment status information, and sends it to the WebSphere Commerce inbound message queue, where the WebSphere Commerce system receives the message. WebSphere Commerce calls the appropriate Controller Command to update the tables ORDSTAT and ORDISTAT with the new order status information.

The Update_WCS_OrderStatus message uses the XML message format and follows `Update_WCS_OrderStatus_20.dtd`.

The following table describes the format of the Update_WCS_OrderStatus message. Each of the four Order Status message types follows the same format, except where noted. The format and the source of the XML element values are described in the following table. For a description of the database column, follow the link to its associated table. All fields are optional unless otherwise noted. The tag value length in the XML message for database fields of type INT, BIGINT and DOUBLE should be CHAR (10), CHAR(19), and CHAR(16) respectively.

Level	Field Name	Comment	Table Name	Column Name	Note
1	*OrderStatusType*	Mandatory	N/A	N/A	OrderConfirm, OrderShipping, OrderInvoice, or OrderStatus

Level	Field Name	Comment	Table Name	Column Name	Note
1.1	SerializationInfo		N/A	N/A	If provided, it is used for checking serialization information for the message.
1.1.1	SequenceNumber		ORDSTAT	OSSEQNUM	
1.1.2	LastUpdate Timestamp		ORDSTAT	OSUPDTIME	
1.2	OrderStatus Header	Mandatory	N/A	N/A	
1.2.A1	Versioning	Attribute	N/A	N/A	Value of 'TRUE' or 'FALSE'. If 'TRUE' then versioning is enabled.
1.2.1	OrderNumber		ORDSTAT	ORDERS_ID/ OSMORDER	If type=ByWCS then ORDERS_ID (which is WebSphere Commerce order reference number), otherwise OSMORDER (which is order reference number generated by back-end system).
	type	Attribute			**ByWCS** or ByBackend
1.2.2	TotalPriceInfo		N/A	N/A	
1.2.2.A1	currency	Attribute	ORDSTAT	OSCPCUR	
1.2.2.1	TotalNetPrice		ORDSTAT	OSPRTOT	
1.2.2.2	TotalTaxPrice		ORDSTAT	OSTXTOT	
1.2.2.3	TotalShippingPrice		ORDSTAT	OSSHTOT	
1.2.2.4	TotalTaxOn ShippingPrice		ORDSTAT	OSSHTXTOT	
1.2.3	Status		ORDSTAT	OSSTATUS	Default values: 'C'='Confirmed' for OrderConfirm 'S'='Shipped' for OrderShipping 'I'='Invoiced' for OrderInvoice
1.2.4	PlacedDate	Mandatory	ORDSTAT	OSPLTIME	
1.2.5	ShippingInfo		N/A	N/A	

Level	Field Name	Comment	Table Name	Column Name	Note
1.2.5.A1	ShipCondition	Attribute	ORDSTAT	OSSCOND	Code to indicate whether partial shipment of order is allowed 'SC'=Ship Complete 'SP'=Ship Partial.
1.2.5.A2	ShipModeFlag	Attribute	ORDSTAT	OSSMFLAG	Code to indicate whether shipping address and shipping mode are at order level or order item level. 'O' = Order level 'I' = Order item level.
1.2.5.1	RequestShipDate		ORDSTAT	OSRSTIME	
1.2.5.2	ScheduledShip Date		ORDSTAT	OSSSTIME	
1.2.5.3	ActualShipDate		ORDSTAT	OSASTIME	
1.2.6	InvoiceInfo				
1.2.6.1	InvoiceDate		ORDSTAT	OSINVTIME	
1.2.6.2	InvoiceValue		ORDSTAT	OSINVVAL	
1.2.7	Comment		ORDSTAT	OSCMNT	
1.2.8	CustomerField	First occurrence	ORDSTAT	FIELD1	
1.2.8	CustomerField	Second occurrence	ORDSTAT	FIELD2	
1.2.8	CustomerField	Third occurrence	ORDSTAT	FIELD3	
1.2.9	UserData		N/A	N/A	
1.3	OrderStatusItem	Repeatable	N/A	N/A	Vector
1.3.A1	Versioning	Attribute	N/A	N/A	Value of 'TRUE' or 'FALSE'. If 'TRUE' then versioning is enabled.
1.3.1	ItemNumber		ORDISTAT	ORDER ITMES_ID/ OIMITEM	If type = ByWCS, then ORDERITMES_ID, if type = ByBackend then OIMITEM
1.3.1.A1	type	Attribute	N/A	N/A	ByWCS or ByBackEnd.
1.3.2	ProductNumber ByMerchant		ORDISTAT	PARTNUMBER	
1.3.3	QuantityInfo		N/A	N/A	
1.3.3.1	RequestedQuantity		ORDISTAT	OIQTREQUEST	
1.3.3.2	ConfirmedQuantity		ORDISTAT	OIQTCONFIRM	

Level	Field Name	Comment	Table Name	Column Name	Note
1.3.3.3	ShippedQuantity		ORDISTAT	OIQTSHIP	
1.3.4	ItemUnitPrice		ORDISTAT	OIUNPRC	
1.3.5	TotalPriceInfo		N/A	N/A	
1.3.5.A1	currency	Attribute	ORDISTAT	OICPCUR	
1.3.5.1	TotalNetPrice		ORDISTAT	OIPRTOT	
1.3.5.2	TotalTaxPrice		ORDISTAT	OITXTOT	
1.3.5.3	TotalShippingPrice		ORDISTAT	OISHTOT	
1.3.5.4	TotalTaxOn ShippingPrice		ORDISTAT	OISHTXTOT	
1.3.6	Status		ORDISTAT	OISTATUS	
1.3.7	PlacedDate		ORDISTAT	OIPLTIME	
1.3.8	ShippingInfo		N/A	N/A	
1.3.8A1	ShipCondition		ORDISTAT	OISCOND	Code to indicate whether partial shipment of the line item is allowed. 'SC'=Ship Complete 'SP'=Ship Partial
1.3.8A2	ShipModeFlag		N/A	N/A	
1.3.8.1	Requested ShipDate		ORDISTAT	OIRSTIME	
1.3.8.2	Scheduled ShipDate		ORDISTAT	OISSTIME	
1.3.8.3	ActualShipDate		ORDISTAT	OIASTIME	
1.3.9	InvoiceInfo		N/A	N/A	
1.3.9.1	InvoiceDate		ORDISTAT	OIINVTIME	
1.3.9.2	InvoiceValue		ORDISTAT	OIINVVAL	
1.3.10	Comment		ORDISTAT	OICMNT	
1.3.11	CustomerField1	First occurrence	ORDISTAT	FIELD1	
1.3.11	CustomerField2	Second occurrence	ORDISTAT	FIELD2	
1.3.11	CustomerField3	Third occurrence	ORDISTAT	FIELD3	
1.3.12	UserData		N/A	N/A	

Order Status Update message

The Order Status Update message is an inbound message that contains status information for a WebSphere Commerce order. A back-end application generates this message and sends it to the WebSphere Commerce inbound message queue, where the WebSphere Commerce system receives the message. WebSphere Commerce uses the OrderStatus command to update the tables ORDSTAT and ORDISTAT with the new order status information.

The Order Status Update message supports two sets of application data: Order Status Update Version 01 and Order Status Update Version 02. Version 02 includes a superset of the data within Version 01.

The Order Status Update message uses the WebSphere Commerce message format and consists of a set of records, which follow each other sequentially in a buffer. The following data describes the Order Status Update message:

```
<ECEDOC>
<PROLOG>ORDER_STATUS_UPDATE_PROLOG_DATA</PROLOG>
<HEADER>
    <HDR010>ORDER_STATUS_UPDATE_HDR010_DATA</HDR010>
    <USRLST>
        <DATUSR>DATUSR_DATA</DATUSR>
                    .
                    .DATUSR repeated loop
                    .
        <DATUSR>DATUSR_DATA</DATUSR>
    </USRLST>
</HEADER>
<ITMLST>
    <ITMDAT>
        <ITM010>ORDER_STATUS_UPDATE_ITM010_DATA</ITM010>
        <USRLST>
            <DATUSR>DATUSR_DATA</DATUSR>
                    .
                    .DATUSR repeated loop
                    .
            <DATUSR>DATUSR_DATA</DATUSR>
        </USRLST>
    </ITMDAT>
                    .
                    .ITEM repeated loop
                    .
    <ITMDAT>
        <ITM010>ORDER_STATUS_UPDATE_ITM010_DATA</ITM010>
        <USRLST>
            <DATUSR>DATUSR_DATA</DATUSR>
                    .
                    .DATUSR repeated loop
                    .
            <DATUSR>DATUSR_DATA</DATUSR>
        </USRLST>
    </ITMDAT>
</ITMLST>
</ECEDOC>
```

Notes:

- All records are in sequential order in the buffer. Indentation is used here for readability; it does not appear in the buffer.
- All fields in the data segments are left-justified and padded to the right with spaces in the buffer.

Data Segments for Order Status Update

- ORDER_STATUS_UPDATE_PROLOG_DATA
  Specifies the type of message the application data defines. In this case, the message is Order Status Update.
- ORDER_STATUS_UPDATE_HDR010_DATA
  Specifies order information within the Order Status Update message.

- DATUSR_DATA
  Specifies optional information to be added to the Order Status Update message. DATUSR_DATA appears in the <HDR> and <ITM> sections of this message.
- ORDER_STATUS_UPDATE_ITM010_DATA
  Specifies item or product shipping information within the Order Status Update message.

Update_WCS_ProductPrice message

The Update_WCS_ProductPrice message is an inbound message that contains price information for a product. A back-end application generates this message and sends it to the WebSphere Commerce inbound message queue. When the WebSphere Commerce system receives the message, it runs a controller command. The controller command that is run depends on the type of message. The following table shows the two types of messages that can be sent, along with their associated controller commands.

Product Price Message Type	Controller Command
OfferPrice Update	ProductOfferPriceUpdate
ListPrice Update	ProductListPriceUpdate

The Update_WCS_ProductPrice message uses the XML message format and follows `Update_WCS_ProductPrice_20.dtd`.

OfferPrice Update

The following table describes the format of the OfferPrice Update variant of the Update_WCS_ProductPrice message. For a description of the database column, follow the link to its associated table. All fields are optional unless otherwise noted. The tag value length in the XML message for database fields of type INT, BIGINT and DOUBLE should be CHAR (10), CHAR(19), and CHAR(16) respectively.

Level	XML Element	Comment	Table Name	Column Name
1	OfferPriceInfo		N/A	N/A
1.1	ProductNumberByMerchant	Mandatory	CATENTRY	PARTNUMBER
1.2	MerchantID	Mandatory	CATENTRY	MEMBER_ID
1.2.A1	type	Attribute	N/A	N/A
1.3	Precedence		OFFER	PRECEDENCE
1.4	TradingPositionContainerID		OFFER	TRADEPOSCN_ID
1.5	Currency	Mandatory	OFFERPRICE	CURRENCY
1.6	ItemUnitPrice		OFFERPRICE	PRICE
1.7	StartTimeStamp		OFFER	STARTDATE
1.8	EndTimeStamp		OFFER	ENDDATE
1.9	MinimumQuantity		OFFER	MINIMUMQUANTITY
1.10	MaximumQuantity		OFFER	MAXIMUMQUANTITY
1.11	QuantityUnit		OFFER	QTYUNIT_ID
1.12	Published		OFFER	PUBLISHED
1.13	PriceCustomField	First occurrence	OFFER	FIELD1
1.13	PriceCustomField	Second occurrence	OFFER	FIELD2
1.14	UserData		N/A	N/A

Behavior for OfferPrice Update:

- The currency type (which references CURRENCY in the OFFERPRICE table) is mandatory and must be specified in ISO 4217 format.
- The combination of the part number (which references PARTNUMBER in the CATENTRY table) and the member id (which references MEMBER_ID in the CATENTRY table) will be used to obtain a catalog entry (CATENTRY_ID). This value, along with either the precedence (PRECEDENCE) or the trade position container (TRADEPOSCN_ID), will be used to obtain a product price reference number (OFFER_ID). The product price reference number, along with the currency type, will be used as the key to update a row in the OFFERPRICE table.
- If the product price reference number (OFFER_ID) matches an existing one in the database, but the currency type does not match a currency type for any record with that product price reference number, a new record will be created in the OFFERPRICE table. This allows you to specify prices in different currencies for the same offer.
- If the precedence (PRECEDENCE) is not specified, then the ProductOfferPriceUpdate command locates all previous records that match the values given without the precedence. The maximum of these values is taken and incremented by one. If a previous record does not exist, then the precedence value is set to 1. A new row is inserted in the table OFFERPRICE with the new precedence value. The precedence value must be less than10^{16}. If the maximum value has been reached, then the new update will be rejected.

ListPrice Update

The following table describes the format of the ListPrice Update variant of the Update_WCS_ProductPrice message. For a description of the database column, follow the link to its associated table. All fields are optional unless otherwise noted. The tag value length in the XML message for database fields of type INT, BIGINT and DOUBLE should be CHAR (10), CHAR(19), and CHAR(16) respectively.

1	ListPriceInfo		N/A	N/A
1.1	ProductNumberByMerchant		CATENTRY	PARTNUMBER
1.2	MerchantID		CATENTRY	MEMBER_ID
1.2.A1	type	Attribute	N/A	N/A
1.3	Currency		LISTPRICE	CURRENCY
1.4	ItemUnitPrice		LISTPRICE	LISTPRICE
1.5	UserData		N/A	N/A

Behavior for ListPrice Update:

- The command updates a record in the LISTPRICE table.
- The partNumber, together with memberId are used to get the catalog entry key (CATENTRY_ID in table CATENTRY).
- If the catalog entry matches an existing one in the LISTPRICE table, but the currency type does not match a currency type for any record for that catalog entry, a new record is created in the LISTPRICE table. This allows you to specify prices in different currencies for the same catalog entry.

Create_WCS_ExpectedInventoryRecord message

The Create_WCS_ExpectedInventoryRecord message is an inbound message that contains information for creating an expected inventory record in the WebSphere Commerce database. A fulfillment center application generates this request and sends it to the WebSphere Commerce inbound message queue, where the WebSphere Commerce system receives it. After WebSphere Commerce processes the message, the ExpectedInventoryRecordCreate command is invoked. The command creates a record in the RA table and one RADETAIL record for each RADETAIL component in the message.

Note: If the inbound message does not contain a valid `StoreID` or a valid `ExpectedDate`, the GenericApplicationError viewname is used for error message composition and the ExpectedInventoryRecordCreate command is not invoked. The response message is generated by `GenericApplicationErrorXML.jsp`.

The Create_WCS_ExpectedInventoryRecord message uses the XML message format and follows `Create_WCS_ExpectedInventoryRecord_10.dtd`.

The following table describes the format of the Create_WCS_ExpectedInventoryRecord message. For a description of the database column, follow the link to its associated table. All fields are optional unless otherwise noted. The tag value length in the XML message for database fields of type INT, BIGINT and DOUBLE should be CHAR (10), CHAR(19), and CHAR(16) respectively.

	Field Name	Comment	Table Name	Column Name	Note
1	BackendRaID		N/A	N/A	This can be used in the response message to associate with the original request
2	StoreID	Mandatory	RA	STORE_ID	N/A
3	VendorID	Mandatory	RA	VENDOR_ID	N/A
4	OrderDate	Mandatory	RA	ORDERDATE	N/A
5	ExternalID		RA	EXTERNALID	N/A
6	FulfillmentCenterID	Mandatory	RADETAIL	FFMCENTER_ID	N/A
7	ItemOwnerID	Mandatory	ITEMSPC	MEMBER_ID	N/A
8	ProductSKU	Mandatory	ITEMSPC	PARTNUMBER	ProductSKU along with ItemOwnerID is used to identify the item specification
9	ExpectedDate	Mandatory	RADETAIL	EXPECTEDDATE	ISO 8601 date format
10	QuantityOrdered	Mandatory	RADETAIL	QTYORDERED	N/A
11	Comment		RADETAIL	RADETAILCOMMENT	N/A
12	BackendRaDetailID		N/A	N/A	This can be used in the response message to associate with the original request.

Create_WCS_PickBatch message

The Create_WCS_PickBatch message is an inbound message that contains information for generating a WebSphere Commerce pickbatch. A fulfillment center application generates this request and sends it to the WebSphere Commerce inbound message queue. After WebSphere Commerce processes the message, the PickBatchGenerate controller command is invoked. This command gets a new PICKBATCH_ID through the key manager, selects all rows with a value of `SHIP` in the STATUS column of the ORDRELEASE database table, generates the XML pick ticket and packing list XML for the input `StoreID` and `FulfillmentCenterID`, and saves them to the PICKBATCH and ORDRELEASE tables respectively. The Response_WCS_PickBatch message is sent in response.

Note: If the inbound message does not contain a valid `StoreID`, the GenericApplicationError viewname is used for error message composition and the PickBatchGenerate command is not invoked. The response message is generated by `GenericApplicationErrorXML.jsp`.

The Create_WCS_PickBatch message uses the XML message format and follows `Create_WCS_PickBatch_10.dtd`.

The following table describes the format of the Create_WCS_PickBatch message. For a description of the database column, follow the link to its associated table. All fields are optional unless otherwise noted. The tag value length in the XML message for database fields of type INT, BIGINT and DOUBLE should be CHAR (10), CHAR(19), and CHAR(16) respectively.

Level	Field Name	Comment	Table Name	Column Name	Note
1	BackendPickBatchID		N/A	N/A	This can be used in the Response_WCS_PickBatch message
2	StoreID	Mandatory	ORDERITEMS	STORE_ID	N/A
3	FulfillmentCenterID	Mandatory	ORDERITEMS	FFMCENTER_ID	N/A

Inquire_WCS_PickPackListDetail message

The Inquire_WCS_PickPackListDetail message is an inbound message that requests the pick ticket and packing list information created by an earlier Create_WCS_PickBatch message. A fulfillment center application generates this request and sends it to the WebSphere Commerce inbound message queue. When processed, the inbound message calls the GetPickPackListDetail command, which redirects to PickPackListResult view task on successful completion, or PickPackListErrorView view task on command failure. On successful completion, the command retrieves the pick ticket XML from the PICKBATCH table and the packing list XML from the ORDRELEASE table for the given PICKBATCH_ID. The PickPackListResult.jsp file generates the outbound Report_WCS_PickPackListDetail message including the pick ticket and packing slips in response.

The Inquire_WCS_PickPackListDetail message uses the XML message format and follows `Inquire_WCS_PickPackListDetail_10.dtd`.

The following table describes the format of the Inquire_PickPackListDetail message. For a description of the database column, follow the link to its associated table. All fields are optional unless otherwise noted. The tag value length in the XML message for database fields of type INT, BIGINT and DOUBLE should be CHAR (10), CHAR(19), and CHAR(16) respectively.

Level	XML Element	Comment	Table Name	Column Name	Note
1	PickBatchID	Mandatory	PICKBATCH	PICKBATCH_ID	N/A

Create_WCS_InventoryReceipt message

The Create_WCS_InventoryReceipt message is an inbound message that contains information for creating a WebSphere Commerce inventory record. A fulfillment center application generates this request and sends it to the WebSphere Commerce inbound message queue, where the WebSphere Commerce system receives the message. When the WebSphere Commerce system receives the message, it calls the ReceiptCreate controller command. If the command executes successfully, the Response_WCS_CreateInvReceipt message is sent in response. If enough inventory exists to fulfill an expected inventory record, the record will be closed.

Note: If the inbound message does not contain a valid `StoreID`or a valid `ReceiptDate`, the GenericApplicationError viewname is used for error message composition and the ReceiptCreate command is not invoked. The response message is generated by `GenericApplicationErrorXML.jsp`.

The Create_WCS_InventoryReceipt message uses the XML message format and follows `Create_WCS_InventoryReceipt_10.dtd`.

The following table describes the format of the Create_WCS_InventoryReceipt message. For a description of the database column, follow the link to its associated table. All fields are optional unless otherwise noted. The tag value length in the XML message for database fields of type INT, BIGINT and DOUBLE should be CHAR (10), CHAR(19), and CHAR(16) respectively.

Level	Field Name	Comment	Table Name	Column Name	Note
1	ItemOwnerID	Mandatory	ITEMSPC	MEMBER_ID	N/A
2	ProductNumberbyManufacturer	Mandatory	ITEMSPC	PARTNUMBER	N/A
3	VersionName		ITEMVERSN	VERSIONNAME	Reserved for IBM internal use.
4	StoreID	Mandatory	RECEIPT	STORE_ID	N/A
5	FulfillmentCenterID	Mandatory	RECEIPT	FFMCENTER_ID	N/A
6	VendorID	Mandatory	RECEIPT	VENDOR_ID	N/A
7	Cost	Mandatory	RECEIPT	COST	N/A
8	Currency	Mandatory	RECEIPT	SETCCUR	N/A
9	QTYReceived	Mandatory	RECEIPT	QTYRECEIVED	N/A
10	ReceiptDate	Mandatory	RECEIPT	RECEIPTDATE	ISO 8601 date format
11	WCSRaDetailID		RECEIPT	RADETAIL_ID	An expected inventory record in the RA table. If it does not exist, the receipt is an ad-hoc type
12	ReceiptComment		RECEIPT	COMMENT1	N/A
13	QualityComment		RECEIPT	COMMENT2	N/A

Update_WCS_InventoryReceipt message

The Update_WCS_InventoryReceipt message is an inbound message that contains information for adjusting the inventory for an item. A fulfillment center application generates this message and sends it to the WebSphere Commerce inbound message queue, where the WebSphere Commerce system receives the message. After WebSphere Commerce receives the message, the InventoryAdjust command is invoked.

Note: If the inbound message does not contain a valid `StoreID`, the GenericApplicationError viewname is used for error message composition and the InventoryAdjust command is not invoked. The response message is generated by `GenericApplicationErrorXML.jsp`.

The Update_WCS_InventoryReceipt message uses the XML message format and follows `Update_WCS_InventoryReceipt_10.dtd`.

The following table describes the format of the Update_WCS_InventoryReceipt message. For a description of the database column, follow the link to its associated table. All fields are optional unless otherwise

noted. The tag value length in the XML message for database fields of type INT, BIGINT and DOUBLE should be CHAR (10), CHAR(19), and CHAR(16) respectively.

Level	Field Name	Comment	Table Name	Column Name	Note
1	ItemOwnerID	Mandatory	ITEMSPC	MEMBER_ID	N/A
2	ProductSKU	Mandatory	ITEMSPC	PARTNUMBER	N/A
3	VersionName		ITEMVERSN	VERSIONNAME	Reserved for IBM internal use.
4	StoreID	Mandatory	RECEIPT	STORE_ID	N/A
5	FulfillmentCenterID	Mandatory	RECEIPT	FFMCENTER_ID	N/A
6	Comment		RECEIPT	COMMENT1	N/A
7	QTYAdjusted	Mandatory	INVADJUST	QUANTITY	Can be positive or negative value
8	InvAdjCodeID	Mandatory	INVADJUST	INVADJCODE_ID	N/A

Create_WCS_ShipmentConfirmation message

The Create_WCS_ShipmentConfirmation message is an inbound message that contains information for issuing shipment confirmation for an item. A fulfillment center application generates this request and sends it to the WebSphere Commerce inbound message queue. After WebSphere Commerce processes the message, the ReleaseShipConfirm command is invoked.

On successful completion, the command redirects to the view task ReleaseShipConfirmRedirectView. The Response_WCS_CreateShipConfirm response message is generated by `CreateShipConfirmOK.jsp`. The command updates the required database, changing the fulfillment status of the item to confirm shipment. It gets a new manifest_id from the MANIFEST table through the key manager, and propagates the MANIFEST table with input data. If the command executes successfully and the UpdateManifestStatus is `1`, the default `ReleaseShipNotify.jsp` generates a notification email.

If the command encounters an error, it redirects to the view task ReleaseShipConfirmErrorView. This error view task for MQSeries is implemented by `CreateShipConfirmError.jsp.`

Note: If the inbound message does not contain a valid `ActualShipDate`, the GenericApplicationError viewname is used for error message composition. The response message is generated by `GenericApplicationErrorXML.jsp`.

The Create_WCS_ShipmentConfirmation message uses the XML message format and follows `Create_WCS_ShipmentConfirmation_10.dtd`.

The following table describes the format of the Create_WCS_ShipmentConfirmation message. For a description of the database column, follow the link to its associated table. All fields are optional unless otherwise noted. The tag value length in the XML message for database fields of type INT, BIGINT and DOUBLE should be CHAR (10), CHAR(19), and CHAR(16) respectively.

Level	Field Name	Comment	Table Name	Column Name	Note
1	UpdateManifestStatus		MANIFEST	UPDATEMANIFESTSTATUS	Default 0, an attribute of parent element ShipmentConfirmation
2	ShipModeID	Mandatory	MANIFEST	SHIPMODE_ID	N/A
3	OrderNumber	Mandatory	MANIFEST	ORDERS_ID	N/A
4	OrderReleaseNum	Mandatory	MANIFEST	ORDERRELEASENUM	N/A

Level	Field Name	Comment	Table Name	Column Name	Note
5	PackageID		MANIFEST	PACKAGEID	N/A
6	TrackingID		MANIFEST	TRACKINGID	N/A
7	PickUpRecordID		MANIFEST	PICKUPRECORDID	N/A
8	ActualShipDate	Mandatory	MANIFEST	DATESHIPPED	ISO 8601 date format
9	ShippingCosts	Mandatory	MANIFEST	SHIPPINGCOSTS	N/A
10	Weight	Mandatory	MANIFEST	WEIGHT	N/A
11	measure	Mandatory	MANIFEST	WEIGHTMEASURE	An attribute of Weight
12	currency	Mandatory	MANIFEST	SETCCURR	An attribute of ShippingCosts

The default value for the UpdateManifestStatus flag is `0`. If it is set to `1`, a task command is called to update the manifest status and an e-mail is sent to inform the customer about completion of product shipment.

The JSP file that generates the response message is Response_WCS_CreateShipConfirm.jsp. ReleaseShipNotify.jsp is the default JSP file, generating the email if UpdateManifestStatus flag is set to 1. See Message composition templates for information on JSP file locations.

Create_WCS_Customer message

The Create_WCS_Customer message is an inbound message that contains customer information for a shopper. A back-end application generates this message and sends it to the WebSphere Commerce inbound message queue, where the WebSphere Commerce system receives the message. WebSphere Commerce adds information for a new shopper by calling the UserRegistrationAdd command.

The Create_WCS_Customer message uses the XML message format and follows `Create_WCS_Customer_20.dtd`.

The format for fields mapping to the database fields for this message is similar to the format for the Update_WCS_Customer message.

Update_WCS_Customer message

The Update_WCS_Customer message is an inbound message that contains customer information for a shopper. A back-end application generates this message and sends it to the WebSphere Commerce inbound message queue, where the WebSphere Commerce system receives the message. WebSphere Commerce updates information for a customer by calling the `UserRegistrationUpdate` command.

The following table describes the format of the Update_WCS_Customer message. For a description of the database column, follow the link to its associated table. All fields are optional unless otherwise noted. The tag value length in the XML message for database fields of type INT, BIGINT and DOUBLE should be CHAR (10), CHAR (19), and CHAR (16) respectively.

Level	Field Name	Comment	Table Name	Column Name
1	Registration	Mandatory	N/A	N/A
1.1	LogonInfo	Mandatory	N/A	N/A
1.1.1	LogonID	Mandatory	USERREG	LOGONID
1.1.2	Password		USERREG	LOGONPASSWORD

Level	Field Name	Comment	Table Name	Column Name
1.1.3	VerifyPassword		N/A	N/A
1.2	StatusInfo		N/A	N/A
1.2.1	CustomerStatus		USERREG	STATUS
1.2.2	PasswordExpired		N/A	N/A
1.2.2.A1	value	Attribute	USERREG	PASSWORDEXPIRED
1.3	Challenge		N/A	N/A
1.3.1	Question		USERREG	CHALLENGEQUESTION
1.3.2	Answer		USERREG	CHALLENGEANSWER
2	AddressInfo		N/A	N/A
2.1	AddressID		ADDRESS	ADDRESS_ID
2.2	AddressNickName		ADDRESS	NICKNAME
2.3	AddressType		ADDRESS	ADDRESSTYPE
2.4	PersonName		N/A	N/A
2.4.1	Title		ADDRESS	PERSONTITLE
2.4.2	LastName		ADDRESS	LASTNAME
2.4.3	FirstName		ADDRESS	FIRSTNAME
2.4.4	MiddleName		ADDRESS	MIDDLENAME
2.5	Address		N/A	N/A
2.5.A1	primary	Attribute	ADDRESS	ISPRIMARY
2.5.A2	self	Attribute	ADDRESS	SELFADDRESS
2.5.1	AddressLine	First occurrence	ADDRESS	ADDRESS1
2.5.1	AddressLine	Second occurrence	ADDRESS	ADDRESS2
2.5.1	AddressLine	Third occurrence	ADDRESS	ADDRESS3
2.5.2	City		ADDRESS	CITY
2.5.3	State		ADDRESS	STATE
2.5.4	ZipCode		ADDRESS	ZIPCODE
2.5.5	Country		ADDRESS	COUNTRY
2.6	ContactInfo		N/A	N/A
2.6.1	Telephone	First occurrence	ADDRESS	PHONE1
2.6.1	Telephone	Second occurrence	ADDRESS	PHONE2
2.6.1.A1	type	Attribute	ADDRESS	PHONE1TYPE/ PHONE2TYPE
2.6.1.A2	publish	Attribute	ADDRESS	PUBLISH PHONE1 / PUBLISH PHONE2
2.6.2	BestCallingTime		ADDRESS	BESTCALLINGTIME
2.6.3	Fax	First occurrence	ADDRESS	FAX1
2.6.3	Fax	Second occurrence	ADDRESS	FAX2
2.6.4	Email	First occurrence	ADDRESS	EMAIL1
2.6.4	Email	Second occurrence	ADDRESS	EMAIL2
2.7	Billing		N/A	N/A
2.7.1	Code		ADDRESS	BILLINGCODE
2.7.2	CodeType		ADDRESS	BILLINGCODETYPE

Level	Field Name	Comment	Table Name	Column Name
2.8	PackageSuppression		ADDRESS	PACKAGESUPPRESSION
2.9	AddressField	First occurrence	ADDRESS	FIELD1
2.9	AddressField	Second occurrence	ADDRESS	FIELD2
2.9	AddressField	Third occurrence	ADDRESS	FIELD3
3	Profile		N/A	N/A
3.A1	type	Attribute	USERS	PROFILETYPE
3.1	Personal		N/A	N/A
3.1.1	DistinguishedName		USERS	DN
3.1.2	PreferredCurrency		USERS	SETCURR
3.1.3	PreferredLanguage		USERS	LANGUAGE_ID
3.1.4	UserField	First occurrence	USERS	FIELD1
3.1.4	UserField	Second occurrence	USERS	FIELD2
3.1.4	UserField	Third occurrence	USERS	FIELD3
3.1.5	DisplayName		USERPROF	DISPLAYNAME
3.1.6	Photo		USERPROF	PHOTO
3.1.7	PreferredMeasure		USERPROF	PREFERREDMEASURE
3.1.8	PreferredCommunication		USERPROF	PREFERREDCOMM
3.1.9	PreferredDelivery		USERPROF	PREFERREDDELIVERY
3.1.10	Description		USERPROF	DESCRIPTION
3.1.14	UserProfileField	First occurrence	USERPROF	FIELD1
3.1.14	UserProfileField	Second occurrence	USERPROF	FIELD2
3.2	Business		N/A	N/A
3.2.1	BusinessTitle		ADDRESS	BUSINESSTITLE
3.2.2	Organization		N/A	N/A
3.2.2.1	OrganizationID		BUSPROF	ORG_ID
3.2.2.2	OrganizationName		ADDRESS	ORGNAME
3.2.2.3	OrganizationUnitId		BUSPROF	ORGUNIT_ID
3.2.2.4	OrganizationUnitName		ADDRESS	ORGUNITNAME
3.2.3	Employee		N/A	N/A
3.2.3.1	EmployeeID		BUSPROF	EMPLOYEEID
3.2.3.2	AlternateID		BUSPROF	ALTERNATEID
3.2.3.1	EmployeeType		BUSPROF	EMPLOYEETYPE
3.2.4	OfficeAddress		ADDRESS	OFFICEADDRESS
3.2.6	DepartmentNumber		BUSPROF	DEPARTMENTNUM
3.2.7	Manager		BUSPROF	MANAGER
3.2.8	Secretary		BUSPROF	SECRETARY
3.3	Demographics		N/A	N/A
3.3.1	Age		USERDEMO	AGE
3.3.2	Gender		USERDEMO	GENDER
3.3.3	Income		USERDEMO	INCOME
3.3.3.A1	Currency	Attribute	USERDEMO	INCOMECURRENCY

Level	Field Name	Comment	Table Name	Column Name
3.3.4	MaritalStatus		USERDEMO	MARITALSTATUS
3.3.5	Children		USERDEMO	CHILDREN
3.3.6	Household		USERDEMO	HOUSEHOLD
3.3.7	CompanyName		USERDEMO	COMPANYNAME
3.3.8	Hobbies		USERDEMO	HOBBIES
3.3.9	OrderBefore		USERDEMO	ORDERBEFORE
3.3.10	TimeZone		USERDEMO	TIMEZONE
3.3.11	DemographicField	First occurrence	USERDEMO	FIELD1
3.3.11	DemographicField	Second occurrence	USERDEMO	FIELD2
3.3.11	DemographicField	Third occurrence	USERDEMO	FIELD3
3.3.11	DemographicField	Fourth occurrence	USERDEMO	FIELD4
3.3.11	DemographicField	Fifth occurrence	USERDEMO	FIELD5
3.3.11	DemographicField	Sixth occurrence	USERDEMO	FIELD6
3.3.11	DemographicField	Seventh occurrence	USERDEMO	FIELD7
4	UserData		N/A	N/A
4.1	UserDataField		N/A	N/A

Update_WCS_ProductInventory message

The Update_WCS_ProductInventory message is an inbound message that contains inventory information for a product. A back-end application generates this message and sends it to the WebSphere Commerce inbound message queue, where the WebSphere Commerce system receives the message. WebSphere Commerce updates the INVENTORY table with the new inventory information.

The Update_WCS_ProductInventory message uses the XML message format and follows `Update_WCS_ProductInventory_20.dtd`.

The format and the source of the XML element values are described in the following table. For a description of the database column, follow the link to its associated table. Fields are optional unless otherwise noted. The tag value length in the XML message for database fields of type INT, BIGINT and DOUBLE should be CHAR (10), CHAR(19), and CHAR(16) respectively.

Level	XML Element	Comment	Table Name	Column Name	Note
1	ProductNumber ByMerchant	Mandatory (see note)	INVENTORY	CATENTRY_ID	Mandatory only when ProductSKU is not used. This attribute should not be used when ProductSKU is used.
2	MerchantID	Mandatory	INVENTORY	STORE_ID	
3	Quantity	Mandatory	INVENTORY	QUANTITY	
4	UserData		N/A	N/A	
5	FulfillmentCenterID		INVENTORY	FFMCENTER_ID	

Level	XML Element	Comment	Table Name	Column Name	Note
6	ProductSKU	Mandatory (see note)	CATENTRY	PARTNUMBER	Mandatory only when ProductNumberByMerchant is not used. This attribute should not be used when ProductNumberByMerchant is used.

Behavior:

- The WebSphere Commerce product reference number (which references CATENTRY_ID in the INVENTORY table) and the merchant reference number (STORE_ID) are used to update a row in the INVENTORY table.
- If the row in the table INVENTORY does not exist, an exception occurs.

Product Price Update message

The Product Price Update message is an inbound message that contains price information for a product. A back-end application generates this message and sends it to the WebSphere Commerce inbound message queue. When the WebSphere Commerce system receives the message, it runs the ProductOfferPriceUpdate command.

The Product Price Update message supports two sets of application data: Product Price Update version 01 and Product Price Update 02. Version 02 includes a superset of the data within version 01.

The Product Price Update message uses the WebSphere Commerce message format and consists of a set of records, which follow each other sequentially in a buffer. The following data describes the Product Price Update message:

```
<ECEDOC>
<PROLOG>PRODUCT_PRICE_UPDATE_PROLOG_DATA</PROLOG>
<HEADER>
    <HDR010>PRODUCT_PRICE_UPDATE_HDR010_DATA</HDR010>
</HEADER>
</ECEDOC>
```

Notes:

- All records are in sequential order in the buffer. Indentation is used here for readability; it does not appear in the buffer.
- All fields in the data segments are left-justified and padded to the right with spaces in the buffer.

Data Segments for Product Price Update

- PRODUCT_PRICE_UPDATE_PROLOG_DATA
  Specifies the type of message the application data defines. In this case, the message is Product Price Update.
- PRODUCT_PRICE_UPDATE_HDR010_DATA
  Specifies item or product pricing information within the Product Price Update message.

Product Quantity Update message

The Product Quantity Update message is an inbound message that contains inventory information for a product. A back-end application generates this message and sends it to the WebSphere Commerce inbound message queue, where the WebSphere Commerce system receives the message.

The Product Quantity Update message uses the WebSphere Commerce message format and consists of a set of records, which follow each other sequentially in a data buffer. The following data describes the Product Quantity Update message:

```
<ECEDOC>
<PROLOG>PRODUCT_QUANTITY_UPDATE_PROLOG_DATA</PROLOG>
<HEADER>
    <HDR010>PRODUCT_QUANTITY_UPDATE_HDR010_DATA</HDR010>
</HEADER>
</ECEDOC>
```

Notes:

- All records are in sequential order in the buffer. Indentation is used here for readability; it does not appear in the buffer.
- All fields in the data segments are left-justified and padded to the right with spaces in the buffer.

Data Segments for Product Quantity Update

- PRODUCT_QUANTITY_UPDATE_PROLOG_DATA
  Specifies the type of message the application data defines. In this case, the message is Product Quantity Update.
- PRODUCT_QUANTITY_UPDATE_HDR010_DATA
  Specifies product information within the Product Quantity Update message.

Order Create message

The Order Create message is an outbound message that contains order details for a completed order. The WebSphere Commerce system generates this message and sends its data to an outbound message queue, where a back-end system receives the message. This application takes the message and continues any back-end business processes required to complete the specified task for the order. Use Order Create to fulfill the order process of orders that begin with the WebSphere Commerce system, but also require some additional or back-end work by a separate system.

The Order Create message uses the legacy message format and consists of a set of records, which follow each other sequentially in a buffer. The message contains order, shopper, billing, merchant, and shipping information. The following data describes the Order Create message:

```
<ECEDOC>
<PROLOG>ORDER_CREATE_PROLOG_DATA</PROLOG>
<HEADER>
    <HDR010>ORDER_CREATE_HDR010_DATA</HDR010>
    <HDR020>ORDER_CREATE_HDR020_DATA</HDR020>
    <HDR030>ORDER_CREATE_HDR030_DATA</HDR030>
    <HDR040>ORDER_CREATE_HDR040_DATA</HDR040>
    <USRLST>
        <DATUSR>DATUSR_DATA</DATUSR>
                    .
                    .DATUSR repeated loop
                    .
        <DATUSR>DATUSR_DATA</DATUSR>
```

```
    </USRLST>
</HEADER>
<ITMLST>
    <ITMDAT>
        <ITM010>ORDER_CREATE_ITM010_DATA</ITM010>
        <USRLST>
            <DATUSR>DATUSR_DATA</DATUSR>
                    .
                    .DATUSR repeated loop
                    .
            <DATUSR>DATUSR_DATA</DATUSR>
        </USRLST>
    </ITMDAT>
                    .
                    .ITEM repeated loop
                    .
    <ITMDAT>
        <ITM010>ORDER_CREATE_ITM010_DATA</ITM010>
        <USRLST>
            <DATUSR>DATUSR_DATA</DATUSR>
                    .
                    .DATUSR repeated loop
                    .
            <DATUSR>DATUSR_DATA</DATUSR>
        </USRLST>
    </ITMDAT>
</ITMLST>
</ECEDOC>
```

Notes:

- All records are in sequential order in the buffer. Indentation is used here for readability; it does not appear in the buffer.
- All fields in the data segments are left-justified and padded to the right with spaces in the buffer.

Data Segments for Order Create

- ORDER_CREATE_PROLOG_DATA
  Specifies the type of message the application data defines. In this case, the message is Order Create.
- ORDER_CREATE_HDR010_DATA
  Specifies order information within the Order Create message.
- ORDER_CREATE_HDR020_DATA
  Specifies shopper information within the Order Create message.
- ORDER_CREATE_HDR030_DATA
  Specifies billing information within the Order Create message.
- ORDER_CREATE_HDR040_DATA
  Specifies merchant information within the Order Create message.
- DATUSR_DATA
  Specifies optional information to be added to the Order Create message. DATUSR_DATA appears in the <HDR> and <ITM> sections of this message.
- ORDER_CREATE_ITM010_DATA
  Specifies item or product shipping information within the Order Create message.

Customer New message

The Customer New message is a legacy format inbound message that contains customer information for a shopper. A back-end application generates this message and sends it to the WebSphere Commerce inbound message queue, where the WebSphere Commerce system receives the message. WebSphere Commerce updates information for a new shopper by calling the `UserRegistrationAdd` controller command.

The following data describes the Customer New message:

```
<ECEDOC>
<PROLOG>CUSTOMER_UPDATE_PROLOG_DATA</PROLOG>
<HEADER>
    <HDR010>CUSTOMER_UPDATE_HDR010_DATA</HDR010>
    <USRLST>
        <DATUSR>DATUSR_DATA</DATUSR>
                    .
                    .DATUSR repeated loop
                    .
        <DATUSR>DATUSR_DATA</DATUSR>
    </USRLST>
</HEADER>
</ECEDOC>
```

Notes:

- All records are in sequential order in the buffer. Indentation is used here for readability; it does not appear in the buffer.
- All fields in the data segments are left-justified and padded to the right with spaces in the buffer.

Data Segments for Customer New

- CUSTOMER_NEW_PROLOG_DATA
  Specifies the type of message the application data defines. In this case, the message is Customer New.
- CUSTOMER_NEW_HDR010_DATA
  Specifies shopper information within the Customer New message.
- DATUSR_DATA
  Specifies optional information to be added to the Customer New message. DATUSR_DATA appears in the <HDR> section of this message.

Customer Update message

The Customer Update message is an inbound message that contains customer information for a shopper. A back-end application generates this message and sends it to the WebSphere Commerce inbound message queue, where the WebSphere Commerce system receives the message. WebSphere Commerce updates information for a registered shopper by calling the `UserRegistrationUpdate` command.

The Customer Update message uses the WebSphere Commerce message format and consists of a set of records, which follow each other sequentially in a buffer. The following data describes the Customer Update message:

```
<ECEDOC>
<PROLOG>CUSTOMER_UPDATE_PROLOG_DATA</PROLOG>
<HEADER>
    <HDR010>CUSTOMER_UPDATE_HDR010_DATA</HDR010>
    <USRLST>
        <DATUSR>DATUSR_DATA</DATUSR>
```

```
                    .
                    .DATUSR repeated loop
                    .
        <DATUSR>DATUSR_DATA</DATUSR>
    </USRLST>
</HEADER>
</ECEDOC>
```

Notes:

- All records are in sequential order in the buffer. Indentation is used here for readability; it does not appear in the buffer.
- All fields in the data segments are left-justified and padded to the right with spaces in the buffer.

Data Segments for Customer Update

- CUSTOMER_UPDATE_PROLOG_DATA
  Specifies the type of message the application data defines. In this case, the message is Customer Update.
- CUSTOMER_UPDATE_HDR010_DATA
  Specifies shopper information within the Customer Update message.
- DATUSR_DATA
  Specifies optional information to be added to the Customer Update message. DATUSR_DATA appears in the <HDR> section of this message.

Update_NC_Customer message

The Update_NC_Customer message is an inbound message that contains customer information for a shopper. A back-end application generates this message and sends it to the WebSphere Commerce inbound message queue, where the WebSphere Commerce system receives the message. WebSphere Commerce updates information for a registered shopper by calling the UserRegistrationUpdate command. Since the message invokes the same WebSphere Commerce controller command, the message is identical to the Create_NC_Customer message.

The Update_NC_Customer message uses the XML message format and follows the `Update_NC_Customer_10.dtd` file.

The following table describes the format of the Update_NC_Customer message. For a description of the database column, follow the link to its associated table. All fields are optional unless otherwise noted. The tag value length in the XML message for database fields of type INT, BIGINT and DOUBLE should be CHAR (10), CHAR(19), and CHAR(16) respectively.

Level	Field Name	Comment	Table Name	Column Name
1	LoginInfo	Mandatory	N/A	N/A
1.1	LoginID	Mandatory	USERREG	LOGONID
1.2	Password		USERREG	LOGONPASSWORD
1.3	VerifyPassword		USERREG	LOGONPASSWORD
2	MerchantID		N/A	N/A
2.A1	type	Attribute	N/A	N/A
3	MethodOfCommunication		USERPROF	PREFERREDCOMM
4	ChallengeQuestion		USERREG	CHALLENGEQUESTION
5	ChallengeAnswer		USERREG	CHALLENGEANSWER
6	ShopperField	First occurrence	USERS	FIELD1

Level	Field Name	Comment	Table Name	Column Name
6	ShopperField	Second occurrence	USERS	FIELD2
6	ShopperField	Third occurrence	USERS	FIELD3
7	ContactPersonName		N/A	N/A
7.1	Title		ADDRESS	PERSONTITLE
7.2	FullName		N/A	N/A
7.3	LastName		ADDRESS	LASTNAME
7.4	FirstName		ADDRESS	FIRSTNAME
7.5	MiddleName		ADDRESS	MIDDLENAME
7.6	AlternateName		ADDRESS	NICKNAME
8	RepCompany		ADDRESS	ORGNAME
9	Address		N/A	N/A
9.1	AddressLine	First occurrence	ADDRESS	ADDRESS1
9.1	AddressLine	Second occurrence	ADDRESS	ADDRESS2
9.1	AddressLine	Third occurrence	ADDRESS	ADDRESS3
9.2	City		ADDRESS	CITY
9.3	State		ADDRESS	STATE
9.4	Zip		ADDRESS	ZIPCODE
9.5	Country		ADDRESS	COUNTRY
10	ContactInfo		N/A	N/A
10.1	Telephone	First occurrence	ADDRESS	PHONE1
10.1	Telephone	Second occurrence	ADDRESS	PHONE2
10.2	Email	First occurrence	ADDRESS	EMAIL1
10.2	Email	Second occurrence	ADDRESS	EMAIL2
10.3	Fax		ADDRESS	FAX1
11	DayPhoneInfo		N/A	N/A
11.1	PhoneInfo		N/A	N/A
11.1.A1	type	Attribute	ADDRESS	PHONE1TYPE
11.1.A2	isListed	Attribute	ADDRESS	PUBLISHPHONE1
12	EveningPhoneInfo		N/A	N/A
12.1	PhoneInfo		N/A	N/A
12.1.A1	type	Attribute	ADDRESS	PHONE2TYPE
12.1.A2	isListed	Attribute	ADDRESS	PUBLISHPHONE2
13	BestTimeToCall		ADDRESS	BESTCALLINGTIME
14	IncludePackageInsert		ADDRESS	PACKAGESUPPRESSION
15	AddressOptField	First occurrence	ADDRESS	FIELD1
15	AddressOptField	Second occurrence	ADDRESS	FIELD2
15	AddressOptField	Third occurrence	ADDRESS	FIELD3
16	Gender		N/A	N/A
16.A1	value	Attribute	USERDEMO	GENDER
17	AgeGroup		USERDEMO	AGE
18	IncomeGroup		USERDEMO	INCOME

Level	Field Name	Comment	Table Name	Column Name
19	MaritalStatus		USERDEMO	MARITALSTATUS
20	NumberOfChildren		USERDEMO	CHILDREN
21	NumberInHouse		USERDEMO	HOUSEHOLD
22	WorkCompany		USERDEMO	COMPANYNAME
23	Interests		USERDEMO	HOBBIES
24	PreviousOrder		USERDEMO	ORDERBEFORE
25	Demographics	First occurrence	USERDEMO	FIELD1
25	Demographics	Second occurrence	USERDEMO	FIELD2
25	Demographics	Third occurrence	USERDEMO	FIELD3
25	Demographics	Fourth occurrence	USERDEMO	FIELD4
25	Demographics	Fifth occurrence	USERDEMO	FIELD5
25	Demographics	Sixth occurrence	USERDEMO	FIELD6
25	Demographics	Seventh occurrence	USERDEMO	FIELD7
26	UserData		N/A	N/A
26.1	UserDataField		N/A	N/A

Update_NC_OrderStatus message

The Update_NC_OrderStatus message is an inbound message that contains status information for a WebSphere Commerce order. A back-end application generates this message and sends it to the WebSphere Commerce inbound message queue, where the WebSphere Commerce system receives the message. WebSphere Commerce OrderStatus command updates the tables ORDSTAT and ORDISTAT with the new order status information.

The Update_NC_OrderStatus message uses the XML message format and follows `Update_NC_OrderStatus_10.dtd`.

The following table describes the format of the Update_NC_OrderStatus message. For a description of the database column, follow the link to its associated table. All fields are optional unless otherwise noted. The tag value length in the XML message for database fields of type INT, BIGINT and DOUBLE should be CHAR (10), CHAR(19), and CHAR(16) respectively.

Level	Field Name	Comment	Table Name	Column Name	Note
1	OrderStatus Header		N/A	N/A	N/A
1.1	OrderNumber ByBackend		ORDSTAT	OSMORDER	Order reference number generated by the back-end system.
1.2	OrderNumber ByNC	Mandatory	ORDSTAT	ORDERS_ID	WebSphere Commerce order reference number. This is a foreign key that references column ORDERS_ID in table ORDERS.

Level	Field Name	Comment	Table Name	Column Name	Note
1.3	OrderNumber ByBuyer		N/A	N/A	N/A
1.4	TotalPriceInfo		N/A	N/A	N/A
1.4.A1	currency	Attribute	ORDSTAT	OSCPCUR	Currency in which the price is expressed. The format of the currency must adhere to ISO 4217 standards.
1.4.1	TotalNetPrice		ORDSTAT	OSPRTOT	Total product price for the order.
1.4.2	TotalTaxPrice		ORDSTAT	OSTXTOT	Total tax for the order.
1.4.3	TotalShippingPrice		ORDSTAT	OSSHTOT	Total shipping charges for the order.
1.4.4	TotalTaxOn ShippingPrice		ORDSTAT	OSSHTXTOT	Total tax on shipping charges for the order.
1.5	RequisitionerID		N/A	N/A	N/A
1.5.A1	type	Attribute	N/A	N/A	N/A
1.6	Status		ORDSTAT	OSSTATUS	Status of the order.
1.7	DateTime Reference		N/A	N/A	N/A
1.7.1	PlacedDate		ORDSTAT	OSPLTIME	Order placed timestamp
1.7.2	PlacedTime		N/A	N/A	N/A
1.7.3	LastUpdateDate		N/A	N/A	N/A
1.7.4	LastUpdateTime		ORDSTAT	OSUPDTIME	Last update timestamp for the order.
1.8	ShipDate Reference		N/A	N/A	N/A
1.8.1	Requested ShipDate		ORDSTAT	OSRSTIME	Requested shipping timestamp.
1.8.2	Scheduled ShipDate		ORDSTAT	OSSSTIME	Scheduled shipping timestamp.
1.8.3	Actual ShipDate		ORDSTAT	OSASTIME	Actual shipping timestamp.
1.9	CustomerField	First occurrence.	ORDSTAT	FIELD1	Reserved for customization.
1.9	CustomerField	Second occurrence.	ORDSTAT	FIELD2	Reserved for customization.

Level	Field Name	Comment	Table Name	Column Name	Note
1.9	CustomerField	Third occurrence.	ORDSTAT	FIELD3	Reserved for customization.
1.10	UserData		N/A	N/A	N/A
2	OrderStatusItem		N/A	N/A	N/A
2.1	OrderNumber ByBackend		ORDISTAT	OSMORDER	Back-end system order number.
2.2	ItemNumber ByBackend		ORDISTAT	OIMITEM	Back-end system order item number.
2.3	OrderNumber ByNC	Mandatory	ORDISTAT	ORDERS_ID	WebSphere Commerce order reference number. This is a foreign key that references column ORDERS_ID in table ORDERS.
2.4	ItemNumber ByNC		ORDISTAT	ORDER ITMES_ID	WebSphere Commerce item reference number. This is a foreign key that references column ORDERITEMS_ID in table ORDERITEMS.
2.5	Quantity		ORDISTAT	OIQTCONFIRM	Quantity of items confirmed.
2.6	ItemUnitPrice		ORDISTAT	OIUNPRC	Unit price for the item.
2.7	TotalPriceInfo		N/A	N/A	N/A
2.7.A1	currency	Attribute	ORDISTAT	OICPCUR	Currency in which the price of the item is expressed. The format of the currency must adhere to ISO 4217 standards.
2.7.1	TotalNetPrice		ORDISTAT	OIPRTOT	Total price for the item.
2.7.2	TotalTaxPrice		ORDISTAT	OITXTOT	Total tax for the item.
2.7.3	TotalShippingPrice		ORDISTAT	OISHTOT	Total shipping charges for the item.
2.7.4	TotalTaxOn ShippingPrice		ORDISTAT	OISHTXTOT	Total tax on the shipping charges for the item.
2.8	Status		ORDISTAT	OISTATUS	Order item status.

Level	Field Name	Comment	Table Name	Column Name	Note
2.9	DateTime Reference		N/A	N/A	N/A
2.9.1	PlacedDate		ORDISTAT	OIPLTIME	Order item placed timestamp.
2.9.2	PlacedTime		N/A	N/A	N/A
2.9.3	LastUpdateDate		N/A	N/A	N/A
2.9.4	LastUpdateTime		N/A	N/A	N/A
2.10	ShipDate Reference		N/A	N/A	N/A
2.10.1	Requested ShipDate		ORDISTAT	OIRSTIME	Requested shipping timestamp.
2.10.2	Scheduled ShipDate		ORDISTAT	OISSDATE	Scheduled shipping timestamp.
2.10.3	Actual ShipDate		ORDISTAT	OIASTIME	Actual shipping timestamp.
2.11	Instruction		ORDISTAT	OICMNT	Comments from the shopper regarding the item ordered. For example, a shopper can include a message with an ordered gift.
2.12	CustomerField	First occurrence.	N/A	N/A	Reserved for customization.
2.12	CustomerField	Second occurrence.	N/A	N/A	Reserved for customization.
2.12	CustomerField	Third occurrence.	N/A	N/A	Reserved for customization.
2.13	UserData		N/A	N/A	N/A

Create_NC_Customer message

The Create_NC_Customer message is an inbound message that contains customer information for a shopper. A back-end application generates this message and sends it to the WebSphere Commerce inbound message queue, where the WebSphere Commerce system receives the message. WebSphere Commerce updates information for a new shopper by calling the UserRegistrationAdd command.

The Create_NC_Customer message uses the XML message format and follows `Create_NC_Customer_10.dtd`.

The format for fields mapping to the database fields for this message is similar to the format for Update_NC_Customer message.

Update_NC_ProductInventory message

The Update_NC_ProductInventory message is an inbound message that contains inventory information for a product. A back-end application generates this message and sends it to the WebSphere Commerce inbound message queue, where the WebSphere Commerce system receives the message. WebSphere Commerce updates the INVENTORY table with the new inventory information.

The Update_NC_ProductInventory message uses the XML message format and follows `Update_NC_ProductInventory_10.dtd`.

Note: It is recommended that you use the Update_WCS_ProductInventory message in place of Update_NC_ProductInventory as it is an improved version. When using the Update_WCS_ProductInventory message, the ProductNumberByMerchant field can be optionally replaced by ProductSKU.

The format and the source of the XML element values are described in the following table. For a description of the database column, follow the link to its associated table. Fields are optional unless otherwise noted. The tag value length in the XML message for database fields of type INT, BIGINT and DOUBLE should be CHAR (10), CHAR(19), and CHAR(16) respectively.

Level	XML Element	Comment	Table Name	Column Name
1	ProductNumber ByMerchant	Mandatory	INVENTORY	CATENTRY_ID
2	MerchantID	Mandatory	INVENTORY	STORE_ID
3	Quantity	Mandatory	INVENTORY	QUANTITY
4	UserData		N/A	N/A

Behavior:

- The WebSphere Commerce product reference number (which references CATENTRY_ID in the INVENTORY table) and the merchant reference number (STORE_ID) are used to update a row in the INVENTORY table.
- If the row in the table INVENTORY does not exist, an exception occurs.

Update_NC_ProductPrice message

The Update_NC_ProductPrice message is an inbound message that contains price information for a product. A back-end application generates this message and sends it to the WebSphere Commerce inbound message queue, where the WebSphere Commerce system receives the message. The system then invokes the ProductOfferPriceUpdate command which updates the OFFERPRICE table with the new price information.

The Update_NC_ProductPrice message uses the XML message format and follows `Update_NC_ProductPrice_10.dtd`.

The format and the source of the XML element values are described in the following table. For a description of the database column, follow the link to its associated table. All fields are optional unless otherwise noted. For database fields of type INT, the message element should be CHAR (12).

Level	XML Element	Comment	Table Name	Column Name
1	ProductNumber ByMerchant	Mandatory	CATENTRY	PARTNUMBER
2	MerchantID	Mandatory	CATENTRY	MEMBER_ID

Level	XML Element	Comment	Table Name	Column Name
3	PriceGroupInfo		N/A	N/A
3.1	Precedence		OFFER	PRECEDENCE
3.2	Requisitioner GroupID		OFFER	TRADEPOSCN_ID
4	Currency	Mandatory	OFFERPRICE	CURRENCY
5	ItemUnitPrice		OFFERPRICE	PRICE
6	Start Timestamp		OFFER	STARTDATE
7	End Timestamp		OFFER	ENDDATE
8	PriceCustom Field		N/A	N/A
9	UserData		N/A	N/A

Report_NC_PurchaseOrder message

The Report_NC_PurchaseOrder message is an outbound message that contains order details for a completed order. The WebSphere Commerce outbound messaging system generates this message using the OrderCreateXML.jsp composition template and sends its data to an outbound message queue, where a back-end system receives the message. This application takes the message and continues any back-end business processes required to complete the specified task for the order. Use Report_NC_PurchaseOrder to fulfill the order process of orders that begin with the WebSphere Commerce system, but also require some additional or back-end work by a separate system.

The Report_NC_PurchaseOrder message uses the XML message format and follows `Report_NC_PO_10.dtd`.

The DTD file consists of a set of XML elements. The message contains order, shopper, billing, merchant, and shipping information. The following data describes `Report_NC_PO_10.dtd`, which is used for the Report_NC_PurchaseOrder message:

The format and the source of the XML element values are described in the following table. For a description of the database column, follow the link to its associated table. All fields are optional unless otherwise noted. The tag value length in the XML message for database fields of type INT, BIGINT and DOUBLE should be CHAR (10), CHAR(19), and CHAR(16) respectively.

Level	XML Element	Comment	Table Name	Column Name
1	ReportPO Header	Mandatory	N/A	N/A
1.1	OrderNumber ByBuyer		N/A	N/A
1.2	OrderNumber ByMerchant		ORDERS	ORMORDER
1.3	OrderNumber ByNC		ORDERS	ORDERS_ID
1.4	DateTime Reference		N/A	N/A
1.4.1	PlacedDate		ORDERS	TIMEPLACED
1.4.2	PlacedTime		ORDERS	TIMEPLACED

Level	XML Element	Comment	Table Name	Column Name
1.4.3	LastUpdate Date		N/A	N/A
1.4.4	LatestUpdate Time		N/A	N/A
1.5	TotalPriceInfo	Mandatory	N/A	N/A
1.5.A1	currency	Attribute	ORDERS	CURRENCY
1.5.1	TotalNet Price	Mandatory	ORDERS	TOTALPRODUCT
1.5.2	TaxInfo		N/A	N/A
1.5.2.1	Monetary Amount	Mandatory	ORDERS	TOTALTAX
1.5.2.1.A1	currency	Attribute	ORDERS	CURRENCY
1.5.2.2	TaxType		N/A	N/A
1.5.2.3	Percentage		N/A	N/A
1.5.2.4	TaxExemption StatusType		N/A	N/A
1.5.2.5	TaxExemption Number		N/A	N/A
1.5.2.6	TaxJurisdiction Code		N/A	N/A
1.5.2.7	TaxJurisdiction CodeType		N/A	N/A
1.5.3	Total ShippingPrice	Mandatory	ORDERS	TOTALSHIPPING
1.5.4	TotalTaxOn ShippingPrice	Mandatory	ORDERS	TOTALTAXSHIPPING
1.6	Instruction		N/A	N/A
1.7	ShipStatus		ORDERS	STATUS
1.8	BillToInfo		N/A	N/A
1.8.1	OrgName		N/A	N/A
1.8.2	Address	Mandatory	N/A	N/A
1.8.2.1	AddressLine	Mandatory; first repeated occurrence	ADDRESS	ADDRESS1
1.8.2.1	AddressLine	second repeated occurrence	ADDRESS	ADDRESS2
1.8.2.1	AddressLine	third repeated occurrence	ADDRESS	ADDRESS3
1.8.2.2	City	Mandatory	ADDRESS	CITY
1.8.2.3	State	Mandatory	ADDRESS	STATE
1.8.2.4	Zip	Mandatory	ADDRESS	ZIPCODE
1.8.2.5	Country	Mandatory	ADDRESS	COUNTRY
1.8.3	ContactPerson Name	Mandatory	N/A	N/A
1.8.3.1	Title		N/A	N/A
1.8.3.2	FullName		N/A	N/A
1.8.3.3	LastName	Mandatory	ADDRESS	LASTNAME
1.8.3.4	FirstName		ADDRESS	FIRSTNAME

Level	XML Element	Comment	Table Name	Column Name
1.8.3.5	MiddleName		ADDRESS	MIDDLENAME
1.8.3.6	AlternateName		ADDRESS	NICKNAME
1.8.4	ContactInfo	Mandatory	N/A	N/A
1.8.4.1	Telephone	first repeated occurrence	ADDRESS	PHONE1
1.8.4.1	Telephone	second repeated occurrence	ADDRESS	PHONE2
1.8.4.2	Email	first repeated occurrence	ADDRESS	EMAIL1
1.8.4.2	Email	second repeated occurrence	ADDRESS	EMAIL2
1.8.4.3	Fax		ADDRESS	FAX1
1.9	MerchantInfo		N/A	N/A
1.9.1	OrgName	Mandatory	STOREENTDS	DISPLAYNAME
1.9.2	OrgID	first repeated occurrence	N/A	N/A
1.9.2.A1	type	attribute	ORDERS	STOREENT_ID
1.9.3	Address	Mandatory	N/A	N/A
1.9.3.1	AddressLine	Mandatory; first repeated occurrence	STADDRESS	ADDRESS1
1.9.3.1	AddressLine	second repeated occurrence	STADDRESS	ADDRESS2
1.9.3.1	AddressLine	third repeated occurrence	STADDRESS	ADDRESS3
1.9.3.2	City	Mandatory	STADDRESS	CITY
1.9.3.3	State	Mandatory	STADDRESS	STATE
1.9.3.4	Zip	Mandatory	STADDRESS	ZIPCODE
1.9.3.5	Country	Mandatory	STADDRESS	COUNTRY
1.9.4	URL		N/A	N/A
1.9.5	Telephone		STADDRESS	PHONE1
1.9.6	ContactPerson Name		N/A	N/A
1.9.6.1	Title		STADDRESS	PERSONTITLE
1.9.6.2	FullName		N/A	N/A
1.9.6.3	LastName	Mandatory	STADDRESS	LASTNAME
1.9.6.4	FirstName		STADDRESS	FIRSTNAME
1.9.6.5	MiddleName		STADDRESS	MIDDLENAME
1.9.6.6	AlternateName		N/A	N/A
1.9.7	ContactInfo	Mandatory	N/A	N/A
1.9.7.1	Telephone	first repeated occurrence	STADDRESS	PHONE1
1.9.7.1	Telephone	second repeated occurrence	STADDRESS	PHONE2
1.9.7.2	Email	first repeated occurrence	STADDRESS	EMAIL1
1.9.7.3	Email	second repeated occurrence	STADDRESS	EMAIL2
1.9.7.4	Fax		STADDRESS	FAX1
1.10	BuyOrgInfo		N/A	N/A
1.10.1	OrgName	Mandatory	ADDRESS	ORGNAME
1.10.2	OrgID	first repeated occurrence;	N/A	N/A
1.10.2.A1	type	Attribute;	N/A	N/A
1.10.3	Address		N/A	N/A

Level	XML Element	Comment	Table Name	Column Name
1.10.3.1	AddressLine	Mandatory; first repeated occurrence	N/A	N/A
1.10.3.1	AddressLine	second repeated occurrence	N/A	N/A
1.10.3.1	AddressLine	third repeated occurrence	N/A	N/A
1.10.3.2	City	Mandatory	N/A	N/A
1.10.3.3	State	Mandatory	N/A	N/A
1.10.3.4	Zip	Mandatory	N/A	N/A
1.10.3.5	Country	Mandatory	N/A	N/A
1.10.4	URL		N/A	N/A
1.10.5	ContactInfo		N/A	N/A
1.10.5.1	Telephone	first repeated occurrence	N/A	N/A
1.10.5.1	Telephone	second repeated occurrence	N/A	N/A
1.10.5.2	Email	first repeated occurrence	N/A	N/A
1.10.5.2	Email	second repeated occurrence	N/A	N/A
1.10.5.3	Fax		N/A	N/A
1.11	ShipToInfo		N/A	N/A
1.11.1	ContactPerson Name	Mandatory	N/A	N/A
1.11.1.1	Title		N/A	N/A
1.11.1.2	FullName	Mandatory	N/A	N/A
1.11.1.3	LastName	Mandatory	N/A	N/A
1.11.1.4	FirstName		N/A	N/A
1.11.1.5	MiddleName		N/A	N/A
1.11.1.6	AlternateName		N/A	N/A
1.11.2	OfficeAddress Line		N/A	N/A
1.11.3	Address	Mandatory	N/A	N/A
1.11.3.1	AdddressLine	Mandatory; first repeated occurrence	N/A	N/A
1.11.3.1	AdddressLine	second repeated occurrence	N/A	N/A
1.11.3.1	AdddressLine	third repeated occurrence	N/A	N/A
1.11.3.2	City	Mandatory	N/A	N/A
1.11.3.3	State	Mandatory	N/A	N/A
1.11.3.4	Zip	Mandatory	N/A	N/A
1.11.3.5	Country	Mandatory	N/A	N/A
1.11.4	ContactInfo		N/A	N/A
1.11.4.1	Telephone	first repeated occurrence	N/A	N/A
1.11.4.1	Telephone	second repeated occurrence	N/A	N/A
1.11.4.2	Email	first repeated occurrence	N/A	N/A
1.11.4.2	Email	second repeated occurrence	N/A	N/A
1.11.4.3	Fax		N/A	N/A
1.11.5	Comment		N/A	N/A
1.12	Requisitioner Info		N/A	N/A

Level	XML Element	Comment	Table Name	Column Name
1.12.1	RequisitionerID	first repeated occurrence	ORDERS	MEMBER_ID
1.12.1.A1	type	Mandatory; attribute	N/A	N/A
1.12.1	RequisitionerID	second repeated occurrence	USERREG	LOGONID
1.12.1.A2	type	Mandatory; attribute	N/A	N/A
1.12.2	Requisitioner Group		N/A	N/A
1.12.3	Address		N/A	N/A
1.12.3.1	AddressLine	Mandatory; first repeated occurrence	ADDRESS	ADDRESS1
1.12.3.1	AddressLine	second repeated occurrence	ADDRESS	ADDRESS2
1.12.3.1	AddressLine	third repeated occurrence	ADDRESS	ADDRESS3
1.12.3.2	City	Mandatory	ADDRESS	CITY
1.12.3.3	State	Mandatory	ADDRESS	STATE
1.12.3.4	Zip	Mandatory	ADDRESS	ZIPCODE
1.12.3.5	Country	Mandatory	ADDRESS	COUNTRY
1.12.4	ContactPerson Name		N/A	N/A
1.12.4.1	Title		ADDRESS	PERSONTITLE
1.12.4.2	FullName	Mandatory	N/A	N/A
1.12.4.3	LastName	Mandatory	ADDRESS	LASTNAME
1.12.4.4	FirstName		ADDRESS	FIRSTNAME
1.12.4.5	MiddleName		ADDRESS	MIDDLENAME
1.12.4.6	AlternateName		N/A	N/A
1.12.5	ContactInfo	Mandatory	N/A	N/A
1.12.5.1	Telephone	first repeated occurrence	ADDRESS	PHONE1
1.12.5.1	Telephone	second repeated occurrence	ADDRESS	PHONE2
1.12.5.2	Email	first repeated occurrence	ADDRESS	EMAIL1
1.12.5.2	Email	second repeated occurrence	ADDRESS	EMAIL2
1.12.5.3	Fax		ADDRESS	FAX1
1.13	ShipDate Reference		N/A	N/A
1.13.1	Requested ShipDate		N/A	N/A
1.13.2	Scheduled ShipDate		N/A	N/A
1.13.3	Actual ShipDate		N/A	N/A
1.14	PCardInfo		N/A	N/A
1.14.1	Monetary Amount		ORDPAYMTHD	MAXAMOUNT
1.14.1.A1	currency	Attribute	ORDERS	CURRENCY
1.14.2	CardType		ORDPAYMTHD	PAYMETHOD
1.14.3	CardNumber		ORDPAYMTHD	PAYDEVICE

Level	XML Element	Comment	Table Name	Column Name
1.14.4	Expiration Date		ORDPAYMTHIS	ENDDATE
1.14.5	IssueDate		ORDPAYMTHIS	STARTDATE
1.14.6	Credit Authorization Number		N/A	N/A
1.14.7	Customer Reference Number		N/A	N/A
1.15	Shipping CarrierInfo		N/A	N/A
1.15.1	Carrier		N/A	N/A
1.15.2	Method		N/A	N/A
1.16	BuyOrg Accounting Detail		N/A	N/A
1.16.1	Percentage		N/A	N/A
1.16.2	Monetary Amount		N/A	N/A
1.16.2.A1	currency	Attribute	N/A	N/A
1.16.3	BudgetCode	Mandatory	N/A	N/A
1.16.4	Description		N/A	N/A
1.16.5	Calculation Code	Mandatory	N/A	N/A
1.17.2	OrderCustomer Field	first repeated occurrence	ORDERS	FIELD1
1.17.2	OrderCustomer Field	first repeated occurrence	ORDERS	FIELD2
1.17.2	OrderCustomer Field	second repeated occurrence	ORDERS	FIELD3
1.18	UserData		N/A	N/A
2	ReportPOItem	Mandatory; first repeated occurrence	N/A	N/A
2.1	ItemLine Number		N/A	N/A
2.2	ItemNumber ByNC		ORDERITEMS	ORDERITEMS_ID
2.3	ProductNumber ByBuyer		N/A	N/A
2.4	ProductNumber ByMerchant	Mandatory	CATENTRY	PARTNUMBER
2.5	Manufacturer Name		N/A	N/A
2.6	Manufacturer URL		N/A	N/A
2.7	Manufacturer PartNumber		N/A	N/A
2.8	ItemUnitPrice	Mandatory	ORDERITEMS	PRICE

Level	XML Element	Comment	Table Name	Column Name
2.8.A1	currency	Attribute	ORDERITEMS	CURRENCY
2.9	TaxInfo	first repeated occurrence	N/A	N/A
2.9.1	Monetary Amount	first repeated occurrence	N/A	N/A
2.9.1.A1	currency	attribute	N/A	N/A
2.9.2	TaxType		N/A	N/A
2.9.3	Percentage		N/A	N/A
2.9.4	TaxExemption StatusType		N/A	N/A
2.9.5	TaxExemption Number		N/A	N/A
2.9.6	TaxJurisdiction Code		N/A	N/A
2.9.7	TaxJurisdiction CodeType		N/A	N/A
2.10	ItemProduct Quantity	Mandatory	ORDERITEMS	QUANTITY
2.11	UnitOf Measure		N/A	N/A
2.12	Classification		N/A	N/A
2.13	ItemProduct ShortDescription		CATENTDESC	SHORTDESCRIPTION
2.14	Instruction		N/A	N/A
2.15	ShipToInfo		N/A	N/A
2.15.1	ContactPerson Name	Mandatory	N/A	N/A
2.15.1.1	Title		N/A	N/A
2.15.1.2	FullName	Mandatory	N/A	N/A
2.15.1.3	LastName	Mandatory	ADDRESS	LASTNAME
2.15.1.4	FirstName		ADDRESS	FIRSTNAME
2.15.1.5	MiddleName		ADDRESS	MIDDLENAME
2.15.1.6	AlternateName		ADDRESS	NICKNAME
2.15.2	Office AddressLine	not supported for this version	N/A	N/A
2.15.3	Address		N/A	N/A
2.15.3.1	AddressLine	Mandatory; first repeated occurrence	ADDRESS	ADDRESS1
2.15.3.1	AddressLine	second repeated occurrence	ADDRESS	ADDRESS2
2.15.3.1	AddressLine	third repeated occurrence	ADDRESS	ADDRESS3
2.15.3.2	City	Mandatory	ADDRESS	CITY
2.15.3.3	State	Mandatory	ADDRESS	STATE
2.15.3.4	Zip	Mandatory	ADDRESS	ZIPCODE
2.15.3.5	Country	Mandatory	ADDRESS	COUNTRY
2.15.4	ContactInfo	Mandatory	N/A	N/A
2.15.4.1	Telephone	first repeated occurrence	ADDRESS	PHONE1

Level	XML Element	Comment	Table Name	Column Name
2.15.4.1	Telephone	second repeated occurrence	ADDRESS	PHONE2
2.15.4.2	Email	first repeated occurrence	ADDRESS	EMAIL1
2.15.4.2	Email	second repeated occurrence	ADDRESS	EMAIL2
2.15.4.3	Fax		ADDRESS	FAX1
2.15.5	Comment		ORDERITEMS	COMMENTS
2.16	Shipping CarrierInfo		N/A	N/A
2.16.1	Carrier	Mandatory	SHIPMODE	CARRIER
2.16.2	Method	Mandatory	SHIPMODE	CODE
2.17	ShipStatus		ORDERITEMS	STATUS
2.18	DateTime Reference	Mandatory	N/A	N/A
2.18.1	PlacedDate	Mandatory	ORDERITEMS	LASTCREATE
2.18.2	PlacedTime		ORDERITEMS	LASTCREATE
2.18.3	LastUpdate Date		ORDERITEMS	LASTUPDATE
2.18.4	LastUpdate Time		ORDERITEMS	LASTUPDATE
2.19	Product Measurement		N/A	N/A
2.20	BuyOrg Accounting Detail	first repeated occurrence	N/A	N/A
2.20.1	Percentage		N/A	N/A
2.20.2	Monetary Amount		N/A	N/A
2.20.2.A1	currency	Attribute	N/A	N/A
2.20.3	BudgetCode	Mandatory	N/A	N/A
2.20.4	Description		N/A	N/A
2.20.5	Calculation Code	Mandatory	N/A	N/A
2.21	Service Allowance Charge	first repeated occurrence	N/A	N/A
2.21.1	Allowance ChargeCode	Mandatory	N/A	N/A
2.21.2	Percentage		N/A	N/A
2.21.2	Monetary Amount		N/A	N/A
2.21.2.A1	currency	Attribute	N/A	N/A
2.21.3	Description		N/A	N/A
2.21.4	Calculation Code	Mandatory	N/A	N/A
2.22	ItemShipping Schedule		N/A	N/A

Level	XML Element	Comment	Table Name	Column Name
2.22.1	Quantity	Mandatory	N/A	N/A
2.22.2	ShipDate Reference	Mandatory	N/A	N/A
2.22.2.1	Requested ShipDate		N/A	N/A
2.22.2.1	Scheduled ShipDate		N/A	N/A
2.22.2.3	ActualShip Date		N/A	N/A
2.23	ItemCustomer Field	first repeated occurrence	ORDERITEMS	FIELD1
2.23	ItemCustomer Field	second repeated occurrence	ORDERITEMS	FIELD2
2.24	UserData		N/A	N/A

CUSTOMER_NEW_HDR010_DATA

The Customer New message includes the CUSTOMER_NEW_HDR010_DATA segment. This data segment consists of shopper information for new shoppers for the Customer New message.

The format and the source of the fields for CUSTOMER_NEW_HDR010_DATA are described in the following table. For field lengths, use the table below. For a description of a database column, follow the link to its associated table.

Field Name	Field Type	Table Name	Column Name	Note
NC_HDR010 Version Number	CHAR (2)	N/A	N/A	Fixed value of 01.
NC_LoginID	CHAR (31)	USERREG	LOGONID	Registered shopper's unique logon ID.
NC_Password	CHAR (12)	USERREG	LOGONPASSWORD	Registered shopper's encrypted logon password.
NC_Verify Password	CHAR (12)	N/A	N/A	Verification of the registered shopper's encrypted logon password. To process the message, the value in this field should be the same as the data in NC_Password.
NC_Merchant RefNumber	CHAR (10)	N/A	N/A	N/A

Field Name	Field Type	Table Name	Column Name	Note
NC_Method Comm	CHAR (2)	USERPROF	PREFERREDCOMM	Shopper's preferred method of communication: E1 - E-mail or URL address 1 E2 - E-mail or URL address 2 P1 - Phone number 1 P2 - Phone number 2
NC_Challenge Ques	CHAR (250)	USERREG	CHALLENGEQUESTION	Challenge question for verbal confirmation of the shopper's identity.
NC_Challenge Ans	CHAR (250)	USERREG	CHALLENGEANSWER	Answer to the challenge question.
NC_Shopper Field1	CHAR (254)	USERS	FIELD1	Reserved for merchant customization.
NC_Shopper Field2	CHAR (254)	USERS	FIELD2	Reserved for merchant customization.
NC_Title	CHAR (5)	ADDRESS	PERSONTITLE	Shopper's title: Dr Mr Mrs Ms N - Not provided (default)
NC_Last Name	CHAR (30)	ADDRESS	LASTNAME	Shopper's last name.
NC_First Name	CHAR (30)	ADDRESS	FIRSTNAME	Shopper's first name.
NC_Middle Name	CHAR (30)	ADDRESS	MIDDLENAME	Shopper's middle name.
NC_Rep Company	CHAR (80)	ADDRESS	ORGNAME	Company that the shopper represents.
NC_Phone1	CHAR (30)	ADDRESS	PHONE1	Shopper's primary phone number.
NC_Phone2	CHAR (30)	ADDRESS	PHONE2	Shopper's secondary phone number.
NC_Fax	CHAR (30)	ADDRESS	FAX1	Shopper's facsimile number.
NC_Addr1	CHAR (50)	ADDRESS	ADDRESS1	Shopper's address line 1.
NC_Addr2	CHAR (50)	ADDRESS	ADDRESS2	Shopper's address line 2.
NC_Addr3	CHAR (50)	ADDRESS	ADDRESS3	Shopper's address line 3.
NC_City	CHAR (30)	ADDRESS	CITY	Shopper's city name.

Field Name	Field Type	Table Name	Column Name	Note
NC_State	CHAR (20)	ADDRESS	STATE	Shopper's state, province, or equivalent, abbreviated.
NC_Country	CHAR (30)	ADDRESS	COUNTRY	Shopper's country/region name.
NC_ZipCode	CHAR (20)	ADDRESS	ZIPCODE	Shopper's zip code or equivalent.
NC_Email1	CHAR (254)	ADDRESS	EMAIL1	Shopper's primary e-mail or URL address.
NC_Email2	CHAR (254)	ADDRESS	EMAIL2	Shopper's secondary e-mail or URL address.
NC_Day PhoneType	CHAR (3)	ADDRESS	PHONE1TYPE	Type of daytime phone, such as TTY for a teletypewriter for people who have a hearing impairment, or PHN for a standard telephone.
NC_Day PhoneList	CHAR (1)	ADDRESS	PUBLISHPHONE1	1 - Daytime phone number is listed. 0 - Daytime phone number is unlisted.
NC_Evening PhoneType	CHAR (3)	ADDRESS	PHONE2TYPE	Type of evening phone, such as TTY for a teletypewriter for people who have a hearing impairment, or PHN for a standard telephone.
NC_Evening PhoneList	CHAR (1)	ADDRESS	PUBLISHPHONE2	1 - Evening phone number is listed. 0 - Evening phone number is unlisted.
NC_BestTime ToCall	CHAR (1)	ADDRESS	BESTCALLINGTIME	Best time to call indicator: D - Daytime E - Evening
NC_Package Insert	CHAR (1)	ADDRESS	PACKAGESUPPRESSION	Package inserts suppression flag, which indicates the shopper's preference for including package inserts in orders shipped. 1 - Include 0 - Do not include
NC_Address OptField1	CHAR (3)	ADDRESS	FIELD1	Reserved for merchant customization.

Field Name	Field Type	Table Name	Column Name	Note
NC_Address OptField2	CHAR (1)	ADDRESS	FIELD2	Reserved for merchant customization.
NC_Shopper Gender	CHAR (1)	USERDEMO	GENDER	Shopper's gender: F - Female M - Male N - Not provided (default)
NC_Shopper Age	CHAR (10)	USERDEMO	AGE	Shopper's age: 0 - Not provided (default) 1 - 0-9 years 2 - 10-19 years 3 - 20-29 years 4 - 30-39 years 5 - 40-49 years 6 - 50-59 years 7 - 60 years or older
NC_Shopper Income	CHAR (10)	USERDEMO	INCOME	Shopper's annual income: 0 - Not provided (default) 1 - $0 - $19,999 2 - $20,000 - $39,999 3 - $40,000 - $59,999 4 - $60,000 or more
NC_Marital Status	CHAR (1)	USERDEMO	MARITALSTATUS	Shopper's marital status: S - Single M - Married C - Common Law P - Separated D - Divorced W - Widowed 0 - Other N - Not Provided
NC_Number OfChildren	CHAR (10)	USERDEMO	CHILDREN	Number of children. Default is 0.
NC_Number InHouse	CHAR (10)	USERDEMO	HOUSEHOLD	Number of people in the shopper's household. Default is 1.
NC_Shopper Company	CHAR (30)	USERDEMO	COMPANYNAME	The company for which the shopper works.
NC_Shopper Interest	CHAR (254)	USERDEMO	HOBBIES	The shopper's main interests and hobbies.
NC_Previous OrderFlag	CHAR (1)	USERDEMO	ORDERBEFORE	Indicator of whether or not the shopper has previously placed an order.
NC_Demog Field1	CHAR (1)	USERDEMO	FIELD1	Reserved for merchant customization.

Field Name	Field Type	Table Name	Column Name	Note
NC_Demog Field2	CHAR (1)	USERDEMO	FIELD2	Reserved for merchant customization.
NC_Demog Field3	CHAR (1)	USERDEMO	FIELD3	Reserved for merchant customization.
NC_Demog Field4	CHAR (1)	USERDEMO	FIELD4	Reserved for merchant customization.
NC_Demog Field5	CHAR (254)	USERDEMO	FIELD5	Reserved for merchant customization.
NC_Demog Field6	CHAR (10)	USERDEMO	FIELD6	Reserved for merchant customization.

CUSTOMER_UPDATE_HDR010_DATA

The Customer Update message includes the CUSTOMER_UPDATE_HDR010_DATA segment. This data segment consists of shopper information for registered shoppers for the Customer Update message.

The format and the source of the fields for CUSTOMER_UPDATE_HDR010_DATA are described in the following table. For field lengths, use the table below. For a description of a database column, follow the link to its associated table.

Field Name	Field Type	Table Name	Column Name	Note
NC_HDR010 Version Number	CHAR (2)	N/A	N/A	Fixed value of 01.
NC_LoginID	CHAR (31)	USERREG	LOGONID	Registered shopper's unique logon ID.
NC_Password	CHAR (12)	USERREG	LOGONPASSWORD	Registered shopper's encrypted logon password.
NC_Verify Password	CHAR (12)	N/A	N/A	Verification of the registered shopper's encrypted logon password. To process the message, the value in this field should be the same as the data in NC_Password.
NC_Merchant RefNumber	CHAR (10)	N/A	N/A	N/A

Field Name	Field Type	Table Name	Column Name	Note
NC_Method Comm	CHAR (2)	USERPROF	PREFERREDCOMM	Shopper's preferred method of communication: E1 - E-mail or URL address 1 E2 - E-mail or URL address 2 P1 - Phone number 1 P2 - Phone number 2
NC_Challenge Ques	CHAR (250)	USERREG	CHALLENGEQUESTION	Challenge question for verbal confirmation of the shopper's identity.
NC_Challenge Ans	CHAR (250)	USERREG	CHALLENGEANSWER	Answer to the challenge question.
NC_Shopper Field1	CHAR (254)	USERS	FIELD1	Reserved for merchant customization.
NC_Shopper Field2	CHAR (254)	USERS	FIELD2	Reserved for merchant customization.
NC_Title	CHAR (5)	ADDRESS	PERSONTITLE	Shopper's title: Dr Mr Mrs Ms N - Not provided (default)
NC_Last Name	CHAR (30)	ADDRESS	LASTNAME	Shopper's last name.
NC_First Name	CHAR (30)	ADDRESS	FIRSTNAME	Shopper's first name.
NC_Middle Name	CHAR (30)	ADDRESS	MIDDLENAME	Shopper's middle name.
NC_Rep Company	CHAR (80)	ADDRESS	ORGNAME	Company that the shopper represents.
NC_Phone1	CHAR (30)	ADDRESS	PHONE1	Shopper's primary phone number.
NC_Phone2	CHAR (30)	ADDRESS	PHONE2	Shopper's secondary phone number.
NC_Fax	CHAR (30)	ADDRESS	FAX1	Shopper's facsimile number.
NC_Addr1	CHAR (50)	ADDRESS	ADDRESS1	Shopper's address line 1.
NC_Addr2	CHAR (50)	ADDRESS	ADDRESS2	Shopper's address line 2.
NC_Addr3	CHAR (50)	ADDRESS	ADDRESS3	Shopper's address line 3.
NC_City	CHAR (30)	ADDRESS	CITY	Shopper's city name.

Field Name	Field Type	Table Name	Column Name	Note
NC_State	CHAR (20)	ADDRESS	STATE	Shopper's state, province, or equivalent, abbreviated.
NC_Country	CHAR (30)	ADDRESS	COUNTRY	Shopper's country/region name.
NC_ZipCode	CHAR (20)	ADDRESS	ZIPCODE	Shopper's zip code or equivalent.
NC_Email1	CHAR (254)	ADDRESS	EMAIL1	Shopper's primary e-mail or URL address.
NC_Email2	CHAR (254)	ADDRESS	EMAIL2	Shopper's secondary e-mail or URL address.
NC_Day PhoneType	CHAR (3)	ADDRESS	PHONE1TYPE	Type of daytime phone, such as TTY for a teletypewriter for people who have a hearing impairment, or PHN for a standard telephone.
NC_Day PhoneList	CHAR (1)	ADDRESS	PUBLISHPHONE1	1 - Daytime phone number is listed. 0 - Daytime phone number is unlisted.
NC_Evening PhoneType	CHAR (3)	ADDRESS	PHONE2TYPE	Type of evening phone, such as TTY for a teletypewriter for people who have a hearing impairment, or PHN for a standard telephone.
NC_Evening PhoneList	CHAR (1)	ADDRESS	PUBLISHPHONE2	1 - Evening phone number is listed. 0 - Evening phone number is unlisted.
NC_BestTime ToCall	CHAR (1)	ADDRESS	BESTCALLINGTIME	Best time to call indicator: D - Daytime E - Evening
NC_Package Insert	CHAR (1)	ADDRESS	PACKAGESUPPRESSION	Package inserts suppression flag, which indicates the shopper's preference for including package inserts in orders shipped. 1 - Include 0 - Do not include
NC_Address OptField1	CHAR (3)	ADDRESS	FIELD1	Reserved for merchant customization.

Field Name	Field Type	Table Name	Column Name	Note
NC_Address OptField2	CHAR (1)	ADDRESS	FIELD2	Reserved for merchant customization.
NC_Shopper Gender	CHAR (1)	USERDEMO	GENDER	Shopper's gender: F - Female M - Male N - Not provided (default)
NC_Shopper Age	CHAR (10)	USERDEMO	AGE	Shopper's age: 0 - Not provided (default) 1 - 0-9 years 2 - 10-19 years 3 - 20-29 years 4 - 30-39 years 5 - 40-49 years 6 - 50-59 years 7 - 60 years or older
NC_Shopper Income	CHAR (10)	USERDEMO	INCOME	Shopper's annual income: 0 - Not provided (default) 1 - $0 - $19,999 2 - $20,000 - $39,999 3 - $40,000 - $59,999 4 - $60,000 or more
NC_Marital Status	CHAR (1)	USERDEMO	MARITALSTATUS	Shopper's marital status: S - Single M - Married C - Common Law P - Separated D - Divorced W - Widowed 0 - Other N - Not Provided
NC_Number OfChildren	CHAR (10)	USERDEMO	CHILDREN	Number of children. Default is 0.
NC_Number InHouse	CHAR (10)	USERDEMO	HOUSEHOLD	Number of people in the shopper's household. Default is 1.
NC_Shopper Company	CHAR (30)	USERDEMO	COMPANYNAME	The company for which the shopper works.
NC_Shopper Interest	CHAR (254)	USERDEMO	HOBBIES	The shopper's main interests and hobbies.
NC_Previous OrderFlag	CHAR (1)	USERDEMO	ORDERBEFORE	Indicator of whether or not the shopper has previously placed an order.
NC_Demog Field1	CHAR (1)	USERDEMO	FIELD1	Reserved for merchant customization.

Field Name	Field Type	Table Name	Column Name	Note
NC_Demog Field2	CHAR (1)	USERDEMO	FIELD2	Reserved for merchant customization.
NC_Demog Field3	CHAR (1)	USERDEMO	FIELD3	Reserved for merchant customization.
NC_Demog Field4	CHAR (1)	USERDEMO	FIELD4	Reserved for merchant customization.
NC_Demog Field5	CHAR (254)	USERDEMO	FIELD5	Reserved for merchant customization.
NC_Demog Field6	CHAR (10)	USERDEMO	FIELD6	Reserved for merchant customization.

ORDER_CREATE_HDR010_DATA

The Order Create message includes the ORDER_CREATE_HDR010_DATA segment. This data segment consists of order specifications for the Order Create message.

The format and the source of the fields for ORDER_CREATE_HDR010_DATA are described in the following table. For a description of the database column, follow the link to its associated table.

Field Name	Field Type	Table Name	Column Name	Description
NC_HDR010 Version Number	CHAR (2)	N/A	N/A	Fixed value of 01.
NC_Order RefNumber	CHAR (10)	ORDERS	ORDERS_ID	Unique order reference number, internally generated. This is a primary key.
NC_Order Date	CHAR (8)	ORDERS	TIMEPLACED	Date the order was placed, in the format YYYYMMDD.
NC_Order Time	CHAR (6)	ORDERS	TIMEPLACED	Time the order was placed, in the format HHMMSS.
NC_Currency Type	CHAR (10)	ORDERS	CURRENCY	Currency in which the price is expressed. The format of the currency must adhere to ISO 4217 standards.
NC_Total Price	CHAR (16)	ORDERS	TOTALPRODUCT	Total product price for the order.
NC_Total TaxPrice	CHAR (16)	ORDERS	TOTALTAX	Total sales tax for the order.

Field Name	Field Type	Table Name	Column Name	Description
NC_Total ShippingPrice	CHAR (16)	ORDERS	TOTALSHIPPING	Total shipping charges for the order.
NC_Total TaxShipping Price	CHAR (16)	ORDERS	TOTALTAXSHIPPING	Total tax on shipping charges for the order.
NC_Shopper RefNumber	CHAR (10)	ORDERS	MEMBER_ID	Shopper reference number.
NC_Merchant RefNumber	CHAR (10)	ORDERS	STOREENT_ID	Merchant reference number.
NC_Merchant OrderNumber	CHAR (30)	ORDERS	ORMORDER	Unique order reference number generated by the merchant
NC_BillTo RefNumber	CHAR (10)	ORDERS	ADDRESS_ID	Billing address reference number.
NC_Order CustField1	CHAR (10)	ORDERS	FIELD1	Reserved for merchant customization.
NC_Order CustField2	CHAR (16)	ORDERS	FIELD2	Reserved for merchant customization.
NC_Order CustField3	CHAR (254)	ORDERS	FIELD3	Reserved for merchant customization.

ORDER_STATUS_UPDATE_HDR010_DATA

The Order Status Update message includes the ORDER_STATUS_UPDATE_HDR010_DATA segment. This data segment consists of order specifications for the Order Status Update message.

The format and the source of the fields for ORDER_STATUS_UPDATE_HDR010_DATA are described in the following table. For field lengths, use the table below. For a description of a database column, follow the link to its associated table.

Field Name	Field Type	Table Name	Column Name	Note
NC_HDR010 Version Number	CHAR (2)	N/A	N/A	Fixed value of 01.
NC_Order RefNumber	CHAR (10)	ORDSTAT	ORDERS_ID	WebSphere Commerce order reference number.
NC_Customer OrderNumber	N/A	N/A	N/A	N/A
NC_Currency Type	CHAR (10)	ORDSTAT	OSCPCUR	Currency in which the price is expressed. The format of the currency must adhere to ISO 4217 standards.

Field Name	Field Type	Table Name	Column Name	Note
NC_TotalPrice	CHAR (16)	ORDSTAT	OSPRTOT	Total product price for the order.
NC_Total TaxPrice	CHAR (16)	ORDSTAT	OSTXTOT	Total tax for the order.
NC_Total ShippingPrice	CHAR (16)	ORDSTAT	OSSHTOT	Total shipping charges for the order.
NC_Total TaxShipping Price	CHAR (16)	ORDSTAT	OSSHTXTOT	Total tax on shipping charges for the order.
NC_Shopper LoginID	N/A	N/A	N/A	N/A
NC_Merchant OrderNumber	CHAR (30)	ORDSTAT	OSMORDER	Order reference number generated by the merchant.
NC_Order Status	CHAR (32)	ORDSTAT	OSSTATUS	Order status: P - In pending state C - In past state X - canceled I - Inventory update pending (ship to no longer pending) M - Ready for authentication (ship to passed inventory update)
NC_Schedule ShipDate	CHAR (8)	ORDSTAT	OSSSTIME	Scheduled shipping date, in the format YYYYMMDD.
NC_Actual ShipDate	CHAR (8)	ORDSTAT	OSASTIME	Actual shipping date, in the format YYYYMMDD.
NC_PlaceDate	CHAR (8)	ORDSTAT	OSPLTIME	Placing date, in the format YYYYMMDD.

PRODUCT_PRICE_UPDATE_HDR010_DATA

The Product Price Update message includes the PRODUCT_PRICE_UPDATE_HDR010_DATA segment. This data segment consists of product or item pricing information for the Product Price Update message.

The Product Price Update message supports two sets of application data: Product Price Update version 01 and Product Price Update version 02. Version 02 includes a superset of the data within version 01. Specifically, the PRODUCT_PRICE_UPDATE_HDR010_DATA segment for version 01 contains a value of 01 for the field NC_HDR010VersionNumber; whereas version 02 contains a value of 02 for NC_HDR010VersionNumber. In addition, version 02 also contains an additional field called NC_ProductNumber.

The format and the source of the fields for PRODUCT_PRICE_UPDATE_HDR010_DATA for version 02 are described in the following table. For field lengths, use the table below. For a description of a database column, follow the link to its associated table.

Field Name	Field Type	Table Name	Column Name	Description
NC_HDR010 VersionNumber	CHAR (2)	N/A	N/A	(fixed value of 02)
NC_Product RefNumber	CHAR (10)	OFFER	CATENTRY_ID	The CatalogEntry offered for sale.
NC_Product Price	CHAR (16)	OFFERPRICE	PRICE	Price of the product or item.
NC_Currency Type	CHAR (10)	OFFERPRICE	CURRENCY	Currency in which the price is expressed. The format of the currency must adhere to ISO 4217 standards.
NC_Shopper Group	CHAR (10)	OFFER	TRADEPOSCN_ID	The TradingPositionContainer of which the Offer is a part.
NC_Precedence	CHAR (10)	OFFER	PRECEDENCE	Precedence for this price.
NC_Merchant RefNumber	CHAR (10)	CATENTRY	MEMBER_ID	The reference number that identifies the owner of the Catalog Entry.
NC_ProductNumber	CHAR (64)	CATENTRY	PARTNUMBER	The reference number that identifies the owner of the Catalog Entry
NC_Product PriceRefNum	CHAR (10)	OFFER	OFFER_ID	Product or item price reference number.
NC_Start Timestamp	CHAR (26)	OFFER	STARTDATE	Date that the product or item price becomes effective, in the format YYYYMMDD hh:mm:ss.ssssss. Default is the current date and time.
NC_End Timestamp	CHAR (26)	OFFER	ENDDATE	Date that the product or item price expires, in the format YYYY-MM-DD hh:mm:ss.ssssss. Default is 9999-12-31 23:59:59.999999.
NC_Price CustomField1	N/A	N/A	N/A	Reserved for merchant customization.
NC_Price CustomField2	N/A	N/A	N/A	Reserved for merchant customization.

Behavior for version 02:

- The currency type (which references CURRENCY in the OFFERPRICE table) is mandatory and must be specified in ISO 4217 format.

- If the product price reference number (which references OFFER_ID in the OFFER table) is specified, this value along with the currency type, will be used as the key to update a row in the OFFERPRICE table.
- If the product price reference number (OFFER_ID) is not specified, then the combination of the product reference number (which references CATENTRY_ID in the CATENTRY table) and either the precedence (which references PRECEDENCE in the OFFER table) or the trade position container (which references TRADEPOSCN_ID in the OFFER table) will be used to obtain a product price reference number (OFFER_ID). This value, along with the currency type, will be used as the key to update a row in the OFFERPRICE table.
- If the product reference number (CATENTRY_ID) is not specified, then the combination of the product number (which references PARTNUMBER in the CATENTRY table) and the merchant reference number (which references MEMBER_ID in the CATENTRY table) will be used to obtain a product reference number (CATENTRY_ID). This value, along with either the precedence (PRECEDENCE) or the trade position container (TRADEPOSCN_ID), will be used to obtain a product price reference number (OFFER_ID). The product price reference number, along with the currency type, will be used as the key to update a row in the OFFERPRICE table.
- If the product price reference number (OFFER_ID) matches an existing one in the database, but the currency type does not match a currency type for any record with that product price reference number, a new record will be created in the OFFERPRICE table. This allows you to specify prices in different currencies for the same offer.
- If the precedence (PRECEDENCE) is not specified, then the ProductOfferPriceUpdate command locates all previous records that match the values given without the precedence. The maximum of these values is taken and incremented by one. If a previous record does not exist, then the precedence value is set to 1. A new row is inserted in the table OFFERPRICE with the new precedence value. The precedence value must be less than10^{16}. If the maximum value has been reached, then the new update will be rejected.

The format and the source of the fields for PRODUCT_PRICE_UPDATE_HDR010_DATA for version 01 are described in the following table:

Field Name	Field Type	Table Name	Column Name	Description
NC_HDR010 VersionNumber	CHAR (2)	N/A	N/A	Fixed value of 01.
NC_Product RefNumber	CHAR (10)	OFFER	CATENTRY_ID	The CatalogEntry offered for sale.
NC_Product Price	CHAR (16)	OFFERPRICE	PRICE	Price of the product or item.
NC_Currency Type	CHAR (10)	OFFERPRICE	CURRENCY	Currency in which the price is expressed. The format of the currency must adhere to ISO 4217 standards.
NC_Shopper Group	CHAR (10)	OFFER	TRADEPOSCN_ID	The TradingPositionContainer of which the Offer is a part.
NC_Precedence	CHAR (10)	OFFER	PRECEDENCE	Precedence for this price.
NC_Merchant RefNumber	CHAR (10)	CATENTRY	MEMBER_ID	The reference number that identifies the owner of the Catalog Entry.

Field Name	Field Type	Table Name	Column Name	Description
NC_Product PriceRefNum	CHAR (10)	OFFER	OFFER_ID	Product or item price reference number.
NC_Start Timestamp	CHAR (26)	OFFER	STARTDATE	Date and time that the product or item price becomes effective, in the format YYYY-MM-DD hh:mm:ss:ssssss. Default is the current date and time.
NC_End Timestamp	CHAR (26)	OFFER	ENDDATE	Date and time that the product or item price expires, in the format YYYY-MM-DD hh:mm:ss:ssssss. Default is 9999-12-31 23:59:59.999999.
NC_Price CustomField1	CHAR (30)	N/A	N/A	Reserved for merchant customization.
NC_Price CustomField2	CHAR (1)	N/A	N/A	Reserved for merchant customization.

Behavior for version 01:

- The currency type (which references CURRENCY in the OFFERPRICE table) is mandatory and must be specified in ISO 4217 format.
- If the product price reference number (which references OFFER_ID in the OFFER table) is specified, this value along with the currency type, will be used as the key to update a row in the OFFERPRICE table.
- If the product price reference number (OFFER_ID) is not specified, then the combination of the product reference number (which references CATENTRY_ID in the CATENTRY table) and either the precedence (which references PRECEDENCE in the OFFER table) or the trade position container (which references TRADEPOSCN_ID in the OFFER table) will be used to obtain a product price reference number (OFFER_ID). This value, along with the currency type, will be used as the key to update a row in the OFFERPRICE table.
- If the product price reference number (OFFER_ID) matches an existing one in the database, but the currency type does not match a currency type for any record with that product price reference number, a new record will be created in the OFFERPRICE table. This allows you to specify prices in different currencies for the same offer.
- If the precedence (PRECEDENCE) is not specified, then the ProductOfferPriceUpdate command locates all previous records that match the values given without the precedence. The maximum of these values is taken and incremented by one. If a previous record does not exist, then the precedence value is set to 1. A new row is inserted in the table OFFERPRICE with the new precedence value. The precedence value must be less than10^{16}. If the maximum value has been reached, then the new update will be rejected.

PRODUCT_QUANTITY_UPDATE_HDR010_DATA

The Product Quantity Update message includes the PRODUCT_QUANTITY_UPDATE_HDR010_DATA segment. This data segment includes product or item inventory information for the Product Quantity Update message.

The format and the source of the fields for PRODUCT_QUANTITY_UPDATE_HDR010_DATA are described in the following table. For field lengths, use the table below. For a description of a database column, follow the link to its associated table.

Field Name	Field Type	Table Name	Column Name	Note
NC_HDR010 VersionNumber	CHAR (2)	N/A	N/A	Fixed value of 01.
NC_Product RefNumber	CHAR (10)	INVENTORY	CATENTRY_ID	The CatalogEntry for which the product quantity applies.
NC_Product Quantity	CHAR (10)	INVENTORY	QUANTITY	The product quantity.
NC_Product Number	CHAR (64)	CATENTRY	PARTNUMBER	Part Number of the Catalog Entry.
NC_Merchant RefNumber	CHAR (10)	INVENTORY	STORE_ID	The store for which the product quantity applies.

Behavior:

- The merchant reference number (which references STORE_ID in the INVENTORY table) is mandatory.
- The WebSphere Commerce product reference number (which references CATENTRY_ID in the INVENTORY table) and the merchant reference number (STORE_ID) are used to update a row in the INVENTORY table.
- If the product reference number (CATENTRY_ID) is not present, then the merchant reference number (STORE_ID) is used to obtain the member number (which references MEMBER_ID in the CATENTRY table). The member number must be the same as the owner of the catalog entry. The member number (MEMBER_ID), along with the product number (which references PARTNUMBER in the CATENTRY table) are used to obtain a product reference number (CATENTRY_ID). The product reference number, along with the merchant reference number (STORE_ID) are used to update a row in the INVENTORY table.
- If the row in the table INVENTORY does not exist, an error will occur.

ORDER_CREATE_HDR020_DATA

The Order Create message includes the ORDER_CREATE_HDR020_DATA segment. This data segment consists of shopper specifications for the Order Create message.

The format and the source of the fields for ORDER_CREATE_HDR020_DATA are described in the following table. For field lengths, use the table below. For a description of a database column, follow the link to its associated table.

Field Name	Field Type	Table Name	Column Name	Note
NC_HDR020 Version Number	CHAR (2)	N/A	N/A	Fixed value of 01.

Field Name	Field Type	Table Name	Column Name	Note
NC_Shopper LoginID	CHAR (31)	USERREG	LOGONID	Registered shopper's unique logon ID.
NC_Purchaser LastName	CHAR (30)	ADDRESS	LASTNAME	Purchaser's last name.
NC_Purchaser MiddleName	CHAR (30)	ADDRESS	MIDDLENAME	Purchaser's middle name.
NC_Purchaser FirstName	CHAR (30)	ADDRESS	FIRSTNAME	Purchaser's first name.
NC_Purchaser Addr1	CHAR (50)	ADDRESS	ADDRESS1	Purchaser's address line 1.
NC_Purchaser Addr2	CHAR (50)	ADDRESS	ADDRESS2	Purchaser's address line 2.
NC_Purchaser Addr3	CHAR (50)	ADDRESS	ADDRESS3	Purchaser's address line 3.
NC_Purchaser City	CHAR (30)	ADDRESS	CITY	Purchaser's city name.
NC_Purchaser State	CHAR (20)	ADDRESS	STATE	Purchaser's state, province, or equivalent, abbreviated.
NC_Purchaser Country	CHAR (30)	ADDRESS	COUNTRY	Purchaser's country/region.
NC_Purchaser ZipCode	CHAR (20)	ADDRESS	ZIPCODE	Purchaser's zip code or equivalent.
NC_Purchaser Email1	CHAR (254)	ADDRESS	EMAIL1	Purchaser's primary e-mail or URL address.
NC_Purchaser Email2	CHAR (254)	ADDRESS	EMAIL2	Purchaser's secondary e-mail or URL address.
NC_Purchaser Phone1	CHAR (30)	ADDRESS	PHONE1	Purchaser's primary phone number.
NC_Purchaser Phone2	CHAR (30)	ADDRESS	PHONE2	Purchaser's secondary phone number.
NC_Purchaser Fax	CHAR (30)	ADDRESS	FAX1	Purchaser's facsimile number.
NC_Purchaser CompanyName	CHAR (80)	ADDRESS	ORGNAME	Company that the purchaser represents.
NC_Purchaser Shopper GroupName	CHAR (50)	MBRGRP	MBRGRPNAME	Shopper group that purchaser belongs to.

ORDER_CREATE_HDR030_DATA

The Order Create message includes the ORDER_CREATE_HDR030_DATA segment. This data segment consists of billing details for the Order Create message.

The format and the source of the fields for ORDER_CREATE_HDR030_DATA are described in the following table. For field lengths, use the table below. For a description of a database column, follow the link to its associated table.

Field Name	Field Type	Table Name	Column Name	Note
NC_HDR030 VersionNumber	CHAR (2)	N/A	N/A	Fixed value of 01.
NC_BillTo LastName	CHAR (30)	ADDRESS	LASTNAME	Bill to person's last name.
NC_BillTo MiddleName	CHAR (30)	ADDRESS	MIDDLENAME	Bill to person's middle name.
NC_BillTo FirstName	CHAR (30)	ADDRESS	FIRSTNAME	Bill to person's first name.
NC_BillTo Addr1	CHAR (50)	ADDRESS	ADDRESS1	Bill to person's address line 1.
NC_BillTo Addr2	CHAR (50)	ADDRESS	ADDRESS2	Bill to person's address line 2.
NC_BillTo Addr3	CHAR (50)	ADDRESS	ADDRESS3	Bill to person's address line 3.
NC_BillTo City	CHAR (30)	ADDRESS	CITY	Bill to person's city name.
NC_BillTo State	CHAR (20)	ADDRESS	STATE	Bill to person's state, province, or equivalent, abbreviated.
NC_BillTo Country	CHAR (30)	ADDRESS	COUNTRY	Bill to person's country/region.
NC_BillTo ZipCode	CHAR (20)	ADDRESS	ZIPCODE	Bill to person's zip code or equivalent.
NC_BillTo Email1	CHAR (254)	ADDRESS	EMAIL1	Bill to person's primary e-mail or URL address.
NC_Billto Email2	CHAR (254)	ADDRESS	EMAIL2	Bill to person's secondary e-mail or URL address.
NC_BillTo Phone1	CHAR (30)	ADDRESS	PHONE1	Bill to person's primary phone number.
NC_BillTo Phone2	CHAR (30)	ADDRESS	PHONE2	Bill to person's secondary phone number.
NC_BillTo Fax	CHAR (30)	ADDRESS	FAX1	Bill to person's facsimile number.

ORDER_CREATE_HDR040_DATA

The Order Create message includes the ORDER_CREATE_HDR040_DATA segment. This data segment consists of merchant information for the Order Create message.

The format and the source of the fields for ORDER_CREATE_HDR040_DATA are described in the following table. For field lengths, use the table below. For a description of a database column, follow the link to its associated table.

Field Name	Field Type	Table Name	Column Name	Note
NC_HDR040 Version Number	CHAR (2)	N/A	N/A	Fixed value of 01.
NC_Supplier Name	CHAR (80)	STOREENTDS	DISPLAYNAME	Merchant's company name.
NC_Supplier Addr1	CHAR (50)	STADDRESS	ADDRESS1	Merchant's company address line 1.
NC_Supplier Addr2	CHAR (50)	STADDRESS	ADDRESS2	Merchant's company address line 2.
NC_Supplier Addr3	CHAR (50)	STADDRESS	ADDRESS3	Merchant's company address line 3.
NC_Supplier City	CHAR (30)	STADDRESS	CITY	Merchant's company city name.
NC_Supplier State	CHAR (20)	STADDRESS	STATE	Merchant's company state, province, or equivalent, abbreviated.
NC_Supplier Country	CHAR (30)	STADDRESS	COUNTRY	Merchant's company country/region.
NC_Supplier ZipCode	CHAR (20)	STADDRESS	ZIPCODE	Merchant's company zip code or equivalent.
NC_Supplier Phone1	CHAR (30)	STADDRESS	PHONE1	Merchant's company phone number.
NC_Supplier Contact LastName	CHAR (30)	STADDRESS	LASTNAME	Merchant contact's last name.
NC_Supplier Contact MiddleName	CHAR (30)	STADDRESS	MIDDLENAME	Merchant contact's middle name.
NC_Supplier Contact FirstName	CHAR (30)	STADDRESS	FIRSTNAME	Merchant contact's first name.
NC_Supplier ContactTitle	CHAR (30)	STADDRESS	PERSONTITLE	Merchant contact's title.
NC_Supplier Contact Phone1	CHAR (30)	STADDRESS	PHONE1	Merchant contact's primary phone number.
NC_Supplier Contact Phone2	CHAR (30)	STADDRESS	PHONE2	Merchant contact's secondary phone number.
NC_Supplier Contact Email1	CHAR (254)	STADDRESS	EMAIL1	Merchant contact's primary e-mail or URL address.
NC_Supplier Contact Email2	CHAR (254)	STADDRESS	EMAIL2	Merchant contact's secondary e-mail or URL address.

ORDER_CREATE_ITM010_DATA

The Order Create message includes the ORDER_CREATE_ITM010_DATA segment. This data segment consists of item or product shipping specifications for the Order Create message.

The format and the source of the fields for ORDER_CREATE_ITM010_DATA are described in the following table. For field lengths, use the table below. For a description of a database column, follow the link to its associated table.

Field Name	Field Type	Table Name	Column Name	Note
NC_ITM010 VersionNumber	CHAR (2)	N/A	N/A	Fixed value of 01.
NC_Item LineNumber	CHAR (3)	N/A	N/A (generated as a sequential number)	N/A
NC_Item RefNumber	CHAR (10)	ORDERITEMS	ORDERITEMS_ID	Unique ship to reference number, internally generated. This is a primary key.
NC_Item ProductRefNumber	CHAR (10)	ORDERITEMS	CATENTRY_ID	Item or product reference number. This is not a foreign key.
NC_Item ProductNumber	CHAR (64)	CATENTRY	PARTNUMBER	Item SKU or product number.
NC_Item Product ShortDescription	CHAR (254)	CATENTDESC	SHORTDESCRIPTION	Short description of the item or product, including its name.
NC_Item UnitPrice	CHAR (16)	ORDERITEMS	PRICE	Unit price of the item.
NC_Item CurrencyType	CHAR (10)	ORDERITEMS	CURRENCY	Currency in which the price is expressed. The format of the currency must adhere to ISO 4217 standards.
NC_Item ProductQuantity	CHAR (10)	ORDERITEMS	QUANTITY	Quantity ordered.
NC_Item ShipToAddr RefNum	CHAR (10)	ORDERITEMS	ADDRESS_ID	Address reference number for the shipping address.
NC_Item ShipMode RefNum	CHAR (10)	ORDERITEMS	SHIPMODE_ID	Merchant shipping mode reference number
NC_ItemState	CHAR (1)	ORDERITEMS	STATUS	Order Status: P - In pending state C - In past state X - canceled I - Inventory update pending (ship to no longer pending) M - Ready for authentication (ship to passed inventory update)

Field Name	Field Type	Table Name	Column Name	Note
NC_Item ShipTo LastName	CHAR (30)	ADDRESS	LASTNAME	Ship to customer's last name.
NC_Item ShipTo MiddleName	CHAR (30)	ADDRESS	MIDDLENAME	Ship to customer's middle name.
NC_Item ShipTo FirstName	CHAR (30)	ADDRESS	FIRSTNAME	Ship to customer's first name.
NC_Item ShipToAddr1	CHAR (50)	ADDRESS	ADDRESS1	Ship to customer's address line 1.
NC_Item ShipToAddr2	CHAR (50)	ADDRESS	ADDRESS2	Ship to customer's address line 2.
NC_Item ShipToAddr3	CHAR (50)	ADDRESS	ADDRESS3	Ship to customer's address line 3.
NC_Item ShipToCity	CHAR (30)	ADDRESS	CITY	Ship to customer's city name.
NC_Item ShipToState	CHAR (20)	ADDRESS	STATE	Ship to customer's state, province, or equivalent, abbreviated.
NC_Item ShipToCountry	CHAR (30)	ADDRESS	COUNTRY	Ship to customer's country/region.
NC_Item ShipToZipCode	CHAR (20)	ADDRESS	ZIPCODE	Ship to customer's zip code or equivalent.
NC_Item ShipToEmail1	CHAR (254)	ADDRESS	EMAIL1	Ship to customer's primary e-mail or URL address.
NC_Item ShipToEmail2	CHAR (254)	ADDRESS	EMAIL2	Ship to customer's secondary e-mail or URL address.
NC_Item ShipToPhone1	CHAR (30)	ADDRESS	PHONE1	Ship to customer's primary phone number.
NC_Item ShipToPhone2	CHAR (30)	ADDRESS	PHONE2	Ship to customer's secondary phone number.
NC_Item ShipToFax	CHAR (30)	ADDRESS	FAX1	Ship to customer's facsimile number.
NC_Item ShippingCarrier	CHAR (30)	SHIPMODE	CARRIER	Carrier identifier, such as Federal Express.
NC_Item ShippingMethod	CHAR (30)	SHIPMODE	CODE	Carrier service shipping mode, such as FedEx Express Overnight.
NC_Item ShipToComment	CHAR (254)	ORDERITEMS	COMMENTS	Comments from customer, such as a greeting for a gift.
NC_Item Creation Timestamp	CHAR (64)	ORDERITEMS	LASTCREATE	Date and time the ship to entry was made.

Field Name	Field Type	Table Name	Column Name	Note
NC_Item Update Timestamp	CHAR (64)	ORDERITEMS	LASTUPDATE	Date and time the ship to entry was last updated.
NC_Item CustField1	CHAR (10)	ORDERITEMS	FIELD1	Reserved for merchant customization.
NC_Item CustField2	CHAR (254)	ORDERITEMS	FIELD2	Reserved for merchant customization.

ORDER_STATUS_UPDATE_ITM010_DATA

The Order Status Update message includes the ORDER_STATUS_UPDATE_ITM010_DATA segment. This data segment consists of item or product shipping specifications for the Order Status message.

The Order Status Update message supports two sets of application data: Order Status Update version 01 and Order Status Update version 02. Version 02 includes a superset of the data within version 01. Specifically, the ORDER_STATUS_UPDATE_ITM010_DATA segment for version 01 contains a value of 01 for the field NC_HDR010VersionNumber; whereas version 02 contains a value of 02 for NC_HDR010VersionNumber. In addition, Versions 02 also contains some additional fields not within version 01.

The format and the source of the fields for ORDER_STATUS_UPDATE_ITM010_DATA for version 02 are described in the following table. For field lengths, use the table below. For a description of a database column, follow the link to its associated table.

Field Name	Field Type	Table Name	Column Name	Note
NC_ITM010 VersionNumber	CHAR (2)	N/A	N/A (fixed value of 02)	N/A
NC_Order RefNumber	CHAR (10)	ORDISTAT	ORDERS_ID	WebSphere Commerce order reference number.
NC_Merchant OrderNumber	CHAR (30)	ORDISTAT	OSMORDER	Merchant's order reference number.
NC_Item RefNumber	CHAR (10)	ORDISTAT	ORDERITMES_ID	WebSphere Commerce item reference number.
NC_ItemMerchant RefNumber	CHAR (30)	ORDISTAT	OIMITEM	Merchant item reference number.
NC_Order ItemStatus	CHAR (32)	ORDISTAT	OISTATUS	Order item status.
NC_Item Schedule ShipDate	CHAR (8)	ORDISTAT	OISSTIME	Scheduled shipping date, in the format YYYYMMDD.
NC_Item ActualShipDate	CHAR (8)	ORDISTAT	OIASTIME	Actual shipping date, in the format YYYYMMDD.
NC_Item PlaceDate	CHAR (8)	ORDISTAT	OIPLTIME	Placing date, in the format YYYYMMDD.
NC_Item Quantity	CHAR (10)	ORDISTAT	OIQTCONFIRM	Quantity of items ordered.

Field Name	Field Type	Table Name	Column Name	Note
NC_Item CurrencyType	CHAR (10)	ORDISTAT	OICPCUR	Currency in which the price of the item is expressed. The format of the price must adhere to ISO 4217 standards.
NC_Item UnitPrice	CHAR (16)	ORDISTAT	OIPRTOT	Unit price for the item.
NC_Item TotalPrice	CHAR (16)	ORDISTAT	OITOTPRC	Total price for the item.
NC_Item TotalTaxPrice	CHAR (16)	ORDISTAT	OITXTOT	Total sales price for the item.
NC_ItemTotal ShippingPrice	CHAR (16)	ORDISTAT	OISHTOT	Total shipping price for the item.
NC_ItemTotal TaxShippingPrice	CHAR (16)	ORDISTAT	OISHTXTOT	Total tax on the shipping price for the item.
NC_Item Comment	CHAR (250)	ORDISTAT	OICMNT	Comments from the shopper regarding the item ordered. For example, a shopper can include a greeting message with the ordered gift.

The format and the source of the fields for ORDER_STATUS_UPDATE_ITM010_DATA for version 01 are described in the following table:

Field Name	Field Type	Table Name	Column Name	Description
NC_ITM010 VersionNumber	CHAR (2)	N/A	N/A	Fixed value of 01.
NC_Order RefNumber	CHAR (10)	ORDISTAT	ORDERS_ID	WebSphere Commerce order reference number.
NC_Item RefNumber	CHAR (10)	ORDISTAT	ORDERITMES_ID	WebSphere Commerce item reference number.
NC_Order ItemStatus	CHAR (32)	ORDISTAT	OISTATUS	Order item status.
NC_Item Schedule ShipDate	CHAR (8)	ORDISTAT	OISSTIME	Scheduled shipping date, in the format YYYYMMDD.
NC_Item ActualShipDate	CHAR (8)	ORDISTAT	OIASTIME	Actual shipping date, in the format YYYYMMDD.
NC_Item PlaceDate	CHAR (8)	ORDISTAT	OIPLTIME	Placing date, in the format YYYYMMDD.
NC_Item Quantity	CHAR (10)	ORDISTAT	OIQTCONFIRM	Quantity of items ordered.

NC_Item Comment	CHAR (250)	ORDISTAT	OICMNT	Comments from the shopper regarding the item ordered. For example, a shopper can include a greeting message with the ordered gift.

ORDER_CREATE_PROLOG_DATA

The Order Create message includes the ORDER_CREATE_PROLOG_DATA segment. This data segment identifies the type of message that is being defined; that is, the segment indicates that the message is an Order Create message with a field value of ON (Order New).

The format and the source of the fields for ORDER_CREATE_PROLOG_DATA are described in the following table:

Field Name	Field Type	Field Value
NC_MsgType	CHAR (10)	ON
NC_MsgVersion	CHAR (2)	01
NC_RESERVED	CHAR (10)	Reserved for IBM use.

ORDER_STATUS_UPDATE_PROLOG_DATA

The Order Status Update message includes the ORDER_STATUS_UPDATE_PROLOG_DATA segment. This data segment identifies the type of message that is being defined; that is, the segment indicates that the message is an Order Status Update message with a field value of OS (Order Status).

The Order Status Update message supports two sets of application data: Order Status Update version 01 and Order Status Update version 02. Version 02 includes a superset of the data within version 01. Specifically, the ORDER_STATUS_UPDATE_PROLOG_DATA segment for version 01 contains a value of 01 for the field NC_MsgVersion; whereas version 02 contains a value of 02 for NC_MsgVersion.

The format and the source of the fields for ORDER_STATUS_UPDATE_PROLOG_DATA for version 02 are described in the following table:

Field Name	Field Type	Field Value
NC_MsgType	CHAR (10)	OS
NC_MsgVersion	CHAR (2)	02
NC_RESERVED	CHAR (10)	Reserved for IBM use.

The format and the source of the fields for ORDER_STATUS_UPDATE_PROLOG_DATA for version 01 are described in the following table:

Field Name	Field Type	Field Value
NC_MsgType	CHAR (10)	OS
NC_MsgVersion	CHAR (2)	01
NC_RESERVED	CHAR (10)	Reserved for IBM use.

PRODUCT_PRICE_UPDATE_PROLOG_DATA

The Product Price Update message includes the PRODUCT_PRICE_UPDATE_PROLOG_DATA segment. This data segment identifies the type of message that is being defined; that is, the segment indicates that the message is a Product Price Update message with a field value of PP (Product Price).

The Product Price Update message supports two sets of application data: Product Price Update version 01 and Product Price Update version 02. Version 02 includes a superset of the data within version 01. Specifically, the PRODUCT_PRICE_UPDATE_PROLOG_DATA segment for version 01 contains a value of 01 for the field NC_MsgVersion; whereas version 02 contains a value of 02 for NC_MsgVersion.

The format and the source of the fields for PRODUCT_PRICE_UPDATE_PROLOG_DATA for version 02 are described in the following table:

Field Name	Field Type	Field Value
NC_MsgType	CHAR (10)	PP
NC_MsgVersion	CHAR (2)	02
NC_RESERVED	CHAR (10)	Reserved for IBM use.

The format and the source of the fields for PRODUCT_PRICE_UPDATE_PROLOG_DATA for version 01 are described in the following table:

Field Name	Field Type	Field Value
NC_MsgType	CHAR (10)	PP
NC_MsgVersion	CHAR (2)	01
NC_RESERVED	CHAR (10)	Reserved for IBM use.

PRODUCT_QUANTITY_UPDATE_PROLOG_DATA

The Product Quantity Update message includes the PRODUCT_QUANTITY_UPDATE_PROLOG_DATA segment. This data segment identifies the type of message that is being defined; that is, the segment indicates that the message is a Product Quantity Update message with a field value of PQ (Product Quantity).

The format and the source of the fields for PRODUCT_QUANTITY_UPDATE_PROLOG_DATA are described in the following table:

Field Name	Field Type	Field Value
NC_MsgType	CHAR (10)	PQ
NC_MsgVersion	CHAR (2)	01
NC_RESERVED	CHAR (10)	Reserved for IBM use.

CUSTOMER_NEW_PROLOG_DATA

The Customer New message includes the CUSTOMER_NEW_PROLOG_DATA segment. This data segment identifies the type of message that is being defined; that is, the segment indicates that the message is a Customer Update message with a field value of NC (New Customer).

The format and the source of the fields for CUSTOMER_NEW_PROLOG_DATA are described in the following table:

Field Name	Field Type	Field Value
NC_MsgType	CHAR (10)	NC
NC_MsgVersion	CHAR (2)	01
NC_RESERVED	CHAR (10)	Reserved for IBM use.

CUSTOMER_UPDATE_PROLOG_DATA

The Customer Update message includes the CUSTOMER_UPDATE_PROLOG_DATA segment. This data segment identifies the type of message that is being defined; that is, the segment indicates that the message is a Customer Update message with a field value of UC (Update Customer).

The format and the source of the fields for CUSTOMER_UPDATE_PROLOG_DATA are described in the following table:

Field Name	Field Type	Field Value
NC_MsgType	CHAR (10)	UC
NC_MsgVersion	CHAR (2)	01
NC_RESERVED	CHAR (10)	Reserved for IBM use.

DATUSR_DATA for outbound messages

The outbound message Order Create includes the DATUSR_DATA data segment. Include the DATUSR records in this message to send additional data. You can customize the fields to pass extra data that is not included in the messages.

The format and the source of the fields for DATUSR_DATA are described in the following table:

Field Name	Field Type	Description
NC_FieldName	CHAR (8)	Name of the new field to be added.
NC_FieldLength	CHAR (10)	Length of the NC_FieldValue field.
NC_FieldValue	CHAR (variable)	Text string for the value of the new field.

DATUSR_DATA for inbound messages

The inbound messages Order Status Update, Customer New, and Customer Update, include the DATUSR_DATA data segment. Include the DATUSR records in these messages to receive additional data. You can customize the fields to pass extra data that is not included in the messages. DATUSR_DATA can be repeated multiple times.

The format and the source of the fields for DATUSR_DATA are described in the following table:

Field Name	Field Type	Description
NC_FieldName	CHAR (8)	Name of the new field to be added.
NC_FieldLength	CHAR (10)	Length of the NC_FieldValue field.
NC_FieldValue	CHAR (variable)	Text string for the value of the new field.

NCCustomer_10.mod file

The NCCustomer_10.mod customer common file consists of shopper information for new shoppers. It is used for for both the Create_NC_Customer and Update_NC_Customer messages.

All MOD files are located in the following directory:

2000

drive:\Program Files\WebSphere\CommerceServer\xml\messaging

NT

drive:\WebSphere\CommerceServer\xml\messaging

AIX

/usr/WebSphere/CommerceServer/xml/messaging

Solaris

/opt/WebSphere/CommerceServer/xml/messaging

400

/QIBM/Proddata/WebCommerce/xml/messaging

The format and the source of the XML element values for NCCustomer_10.mod are described in the following table. For a description of the database column, follow the link to its associated table. All fields are optional unless otherwise noted.

Level	XML Element	Comment	Table Name	Column Name
1	LoginInfo	Mandatory	N/A	N/A
1.1	LoginID	Mandatory	USERREG	LOGONID
1.2	Password	Mandatory	USERREG	LOGONPASSWORD
1.3	VerifyPassword	Mandatory	N/A	N/A
2	MerchantID		N/A	N/A
3	MethodOf Communication		USERPROF	PREFERREDCOMM
4	Challenge Question		USERREG	CHALLENGEQUESTION
5	Challenge Answer		USERREG	CHALLENGEANSWER
6	ShopperField	First repeated occurrence	USERS	FIELD1
6	ShopperField	Second repeated occurrence	USERS	FIELD2
7	ContactPerson Name	Mandatory	N/A	N/A
7.1	Title		ADDRESS	PERSONTITLE
7.2	FullName	Not supported in this version	N/A	N/A
7.3	LastName	Mandatory	ADDRESS	LASTNAME
7.4	FirstName		ADDRESS	FIRSTNAME
7.5	MiddleName		ADDRESS	MIDDLENAME

Level	XML Element	Comment	Table Name	Column Name
7.6	AlternateName	Not supported with this version	N/A	N/A
8	RepCompany		ADDRESS	ORGNAME
9	Address	Mandatory	N/A	N/A
9.1	AddressLine	Mandatory; first repeated occurrence	ADDRESS	ADDRESS1
9.1	AddressLine	Second repeated occurrence	ADDRESS	ADDRESS2
9.1	AddressLine	Third repeated occurrence	ADDRESS	ADDRESS3
9.2	City	Mandatory	ADDRESS	CITY
9.3	State	Mandatory	ADDRESS	STATE
9.4	Zip	Mandatory	ADDRESS	ZIPCODE
9.5	Country	Mandatory	ADDRESS	COUNTRY
10	ContactInfo	Mandatory	N/A	N/A
10.1	Telephone	First repeated occurrence	ADDRESS	PHONE1
10.1	Telephone	Second repeated occurrence	ADDRESS	PHONE2
10.2	Email	First repeated occurrence	ADDRESS	EMAIL1
10.2	Email	Second repeated occurrence	ADDRESS	EMAIL2
10.3	Fax		ADDRESS	FAX1
11	DayPhoneInfo		N/A	N/A
11.1	PhoneInfo		N/A	N/A
11.1.A1	type	Attribute	ADDRESS	PHONE1TYPE
11.1.A2	isListed	Attribute	ADDRESS	PUBLISHPHONE1
12	EveningPhone Info		N/A	N/A
12.1	PhoneInfo		N/A	N/A
12.1.A1	type	Attribute	ADDRESS	PHONE2TYPE
12.1.A2	isListed	Attribute	ADDRESS	PUBLISHPHONE2
13	BestTimeToCall		ADDRESS	BESTCALLINGTIME
14	Include PackageInsert		ADDRESS	PACKAGESUPPRESSION
15	Address OptField	First repeated occurrence	ADDRESS	FIELD1
15	Address OptField	First repeated occurrence	ADDRESS	FIELD2
16	Gender		N/A	N/A
16.A1	value	Mandatory; attribute	USERDEMO	GENDER
17	AgeGroup		USERDEMO	AGE
18	IncomeGroup		USERDEMO	INCOME
19	MaritalStatus		N/A	N/A
19.A1	value	Mandatory; attribute	USERDEMO	MARITALSTATUS
20	NumberOf Children		USERDEMO	CHILDREN
21	NumberIn House		USERDEMO	HOUSEHOLD

Level	XML Element	Comment	Table Name	Column Name
22	WorkCompany		USERDEMO	COMPANYNAME
23	Interests		USERDEMO	HOBBIES
24	PreviousOrder		USERDEMO	ORDERBEFORE
25	Demographics	First repeated occurrence	USERDEMO	FIELD1
25	Demographics	Second repeated occurrence	USERDEMO	FIELD2
25	Demographics	Third repeated occurrence	USERDEMO	FIELD3
25	Demographics	Fourth repeated occurrence	USERDEMO	FIELD4
25	Demographics	Fifth repeated occurrence	USERDEMO	FIELD5
25	Demographics	Sixth repeated occurrence	USERDEMO	FIELD6
26	UserData		N/A	N/A

NCCommon.mod - source file

WebSphere Commerce defines all inbound XML messages based on DTD files. Each DTD file consists of several XML elements and the `NCCommon.mod` file. In addition, the Create_NC_Customer_10.dtd and Update_NC_Customer_10.dtd files also include information from the `NCCustomer10.mod` common file.

All MOD and DTD files are located in the following directory:

2000

drive:\Program Files\WebSphere\CommerceServer\xml\messaging

NT

drive:\WebSphere\CommerceServer\xml\messaging

AIX

/usr/WebSphere/CommerceServer/xml/messaging

Solaris

/opt/WebSphere/CommerceServer/xml/messaging

400

/QIBM/Proddata/WebCommerce/xml/messaging

NCCustomer_10.mod - source file

WebSphere Commerce defines all inbound XML messages based on DTD files. Each DTD file consists of several XML elements and the `NCCommon.mod` file. In addition, the `Create_NC_Customer_10.dtd` and `Update_NC_Customer_10.dtd` files also include information from the `NCCustomer10.mod` customer common file.

All MOD and DTD files are located in the following directory:

2000

drive:\Program Files\WebSphere\CommerceServer\xml\messaging

NT

drive:\WebSphere\CommerceServer\xml\messaging

AIX

/usr/WebSphere/CommerceServer/xml/messaging

Solaris

/opt/WebSphere/CommerceServer/xml/messaging

400

/QIBM/Proddata/WebCommerce/xml/messaging

Invoke the messaging system compose method

To invoke the compose() method of the outbound messaging system interface, specify the following parameters:

- viewname: The name of the composition view to be used, as specified in the VIEWNAME column of an existing record in the VIEWREG table. For more information on how the VIEWREG table is used, refer to the *WebSphere Commerce Programmer's Guide*.

  Important: In the record referred to in the VIEWREG table, the values of the INTERFACENAME and CLASSNAME columns must contain the name of the interface and class associated with all WebSphere Commerce messaging system view commands. The name of the interface must be com.ibm.commerce.messaging.viewcommands.MessagingViewCommand. The name of the class must be com.ibm.commerce.messaging.viewcommands.MessagingViewCommandImpl.
- CommandContext: For information on the CommandContext interface or the CommandContextImpl class that implements the interface, refer to the *WebSphere Commerce Programmer's Guide*.
- TypedProperty: The values in the typed property must be strings, or objects that implement the toString() method. For more information on the TypedProperty, refer to the *WebSphere Commerce Programmer's Guide.*

The compose() method runs a view command for each of the transports enabled and assigned to the current message type in the Administration Console. The method performs the following processes:

- It uses the viewname parameter as well as the storeId and device format from each transport, as defined in the Administration Console. These values are used to look up the view command in the VIEWREG table.
- It runs the view command, passing it the values specified in the TypedProperty parameter. When the command is run, the system uses the viewname, storeId, and device format id to look up the JSP template in the PROPERTIES column of the VIEWREG table. The JSP template is run and passed the values in the TypedProperty parameter.
- The JSP composes the message, and it is sent through the appropriate transport when a send method is invoked in the object. Sending may be done using transacted, immediate, or request-reply on the messaging object on which composition was run.

SendXMLOrder command

The SendXMLOrder command is used by the WebSphere Commerce outbound messaging system to send the Report_NC_PurchaseOrder XML message to back-end systems. The command uses a message composition template to generate the XML message, then the outbound messaging system sends it to a back-end system.

Behavior

- The task command is enabled by assigning it to the OrderMessagingCmd interface within OrderProcess command.

- Once enabled, it is called before OrderProcess command finish processing.
- The task command calls the messaging system composition services, which uses the OrderCreateXML.jsp composition template to collect of the necessary order information and build the Report_NC_PurchaseOrder outbound XML message.
- If composition is successful, the command attempts to send the message using the outbound messaging system sending services.

Exception Conditions

The command generates an entry in the error log if an exception is encountered.

Chapter 17. Fulfillment integration messages

WebSphere Commerce provides a mechanism for integration with fulfillment center systems using inbound and outbound messages. Inbound fulfillment integration messages are used to run commands in WebSphere Commerce based on inbound requests received from fulfillment center systems. Outbound messages can be generated by the outbound messaging system in order to update fulfillment center systems with events that have taken place, such as receipt of new stock, or an order shipment. To use fulfillment integration messages, you must have an adapter installed, and have the messaging system configured to receive XML messages.

The messaging system is prepared to send and receive a number of pre-defined messages in XML format. This format offers a high degree of readability, making the messages easy to modify and maintain. For an explanation of each message, refer to the sections on inbound and outbound fulfillment integration messages. You can also add new messages. For new inbound messages, you can associate them with either existing WebSphere Commerce commands, or commands that you have created.

The format of the XML messages consists of a set of XML elements defined within specific DTD files. Each DTD may contain one or more common files, identified by a `.mod` file extension. In addition, each inbound message is associated with a WebSphere Commerce controller command in the `sys_template.xml` message template definition file. All DTD, MOD, and XML files are located in the following directory:

2000

drive:\Program Files\WebSphere\CommerceServer\xml\messaging

NT

drive:\WebSphere\CommerceServer\xml\messaging

AIX

/usr/WebSphere/CommerceServer/xml/messaging

Solaris

/opt/WebSphere/CommerceServer/xml/messaging

400

QIBM/ProdData/WebCommerce/xml/messaging

Notices

Any reference to an IBM licensed program in this document is not intended to state or imply that only IBM's licensed program may be used. Any functionally equivalent product, program, or service that does not infringe any of IBM's intellectual property rights may be used instead of the IBM product, program, or service. Evaluation and verification of operation in conjunction with other products, except those expressly designated by IBM, is the user's responsibility.

IBM may have patents or pending patent applications covering subject matter 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:

Director of Licensing
Intellectual Property & Licensing
North Castle Drive, MD-NC119
Armonk, NY 10504-1785
U.S.A.

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

Lab Director
IBM Canada Ltd. Laboratory
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.

This document may contain examples of data and reports used in daily business operations. To illustrate them as completely as possible, the examples may include the names of individuals, companies, brands, and products. All of these names are fictitious and any similarity to the names and addresses used by an actual business enterprise is entirely coincidental.

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.

This product is based on the SET protocol.

Note to U.S. Government Users — Documentation related to restricted rights — Use, duplication or disclosure is subject to restrictions set forth in GSA ADP Schedule Contract with IBM Corp.

Trademarks

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

AIX	CICS	DB2
DB2 Extenders	Encina	HotMedia
IBM	iSeries	MQSeries

SecureWay 400 VisualAge WebSphere

Blaze Advisor is a trademark of HNC Software, Inc. in the United States, other countries, or both.

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

Lotus and Domino are trademarks of Lotus Development Corporation in the United States, other countries, or both.

Microsoft, Windows, Windows NT, and the Windows logo are trademarks of Microsoft Corporation in the United States, other countries, or both.

Oracle is a registered trademark of Oracle Corporation.

SET and the SET Logo are trademarks owned by SET Secure Electronic Transaction LLC. For further information see http://www.setco.org/aboutmark.html.

UNIX is a registered trademark of The Open Group in the United States and other countries.

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