IBM WebSphere Commerce Payments for Multiplatforms

Programmer's Guide and Reference

Version 3.1

IBM WebSphere Commerce Payments for Multiplatforms

Programmer's Guide and Reference

Version 3.1

Note

Before using this information and the product it supports, be sure to read the general information under Appendix D, "Notices" on page 159.

Second Edition (July, 2002)

This edition applies to Version 3.1.3 of IBM® WebSphere® Commerce Payments and to all subsequent releases and modifications until otherwise indicated in new editions.

Contents

Preface

This book is for programmers who are responsible for developing applications that communicate and interact with the IBM WebSphere Commerce Payments.

Note:
IBM WebSphere Commerce Payments for Multiplatforms (hereafter called WebSphere Commerce Payments) was previously known as IBM WebSphere Payment Manager for Multiplatforms. Starting with version 3.1.3, the payments application was renamed to WebSphere Commerce Payments and references to the product were changed throughout this document. References to the former product may still appear in this document and apply to earlier releases of the product.

Conventions in this book

Table 1. Conventions in this document

Boldface	Indicates the name of the item you need to select, the name of a field, or a string you must enter.
Italics	Indicates book titles or variable information that must be replaced by an actual value.
`Monospace`	Indicates an example, a portion of a file, or a previously entered value.

Additional information

More information is available from these documents and Web sites:

- The *IBM WebSphere Commerce Payments for Multiplatforms Installation Guide Version 3.1* (WebSphere Commerce Payments Installation Guide) provides information for installing, migrating, and starting WebSphere Commerce Payments.
- The *IBM WebSphere Commerce Payments for Multiplatforms Administrator's Guide Version 3.1* (WebSphere Commerce Payments Administrator's Guide) contains conceptual information and shows how to configure WebSphere Commerce Payments using the user interface.
- The *IBM WebSphere Commerce Payments for Multiplatforms, Cassette for SET Supplement Version 3.1* (WebSphere Commerce Payments Cassette for SET Supplement) provides information about using the SET protocol with the WebSphere Commerce Payments, including installation and configuration information.
- The *IBM WebSphere Commerce Payments for Multiplatforms, Cassette for VisaNet Supplement, Version 3.1* (WebSphere Commerce Payments Cassette for VisaNet Supplement) provides information about using WebSphere Commerce Payments to access the VisaNet system, including installation and configuration information.
- The *IBM WebSphere Commerce Payments for Multiplatforms Cassette for CyberCash Supplement, Version 3.1* (WebSphere Commerce Payments Cassette for CyberCash Supplement) provides information about using the WebSphere Commerce Payments to access the CyberCash Cash Register service, including installation and configuration information.

- The *IBM WebSphere Commerce Payments for Multiplatforms, Cassette for BankServACH Version 3.1* (WebSphere Commerce Payments Cassette for BankServACH Supplement) provides information about using WebSphere Commerce Payments to access the Automated Clearing House (ACH) network through the BankServ gateway. Installation and configuration information is included.

The above documents are available after installation of WebSphere Commerce Payments (or the cassette software) through the WebSphere Commerce Payments user interface. On iSeries®, the documentation is also accessible off the iSeries Tasks Page at http://hostname:2001 where hostname is the TCP/IP host name of the iSeries system. The link name in the navigation frame is Documentation.

All documents are available on the WebSphere Commerce Payments CD-ROMs in Portable Document Format (PDF). For the latest Acrobat reader, see: http://www.adobe.com. On iSeries systems, the documentation is compressed into a save file and is only available after WebSphere Commerce Payments has been installed.

You can also reference the following related Web sites for more information:

- **http://www.ibm.com/payment** provides more information on the IBM Payment Suite products
- **http://www.software.ibm.com/commerce/payment/support/serv/index.html** provides current WebSphere Commerce Payments technical information and links to the latest soft copy views of all WebSphere Commerce Payments documentation.
- **http://www.ibm.com/software/websphere/appserv/library.html** provides documentation links for IBM WebSphere Application Server.
- **http://www.ibm.com/software/data/pubs/index.html** provides documentation links for IBM Universal Database.

Part 1. Introduction

Chapter 1. WebSphere Commerce Payments concepts

The IBM WebSphere Commerce Payments provides a generic framework with the capability of supporting different payment methods with protocol-specific *cassettes*. A merchant uses the payment and administration commands to process orders. The WebSphere Commerce Payments translates the generic command into a payment protocol-specific request and forwards it to the appropriate recipient, such as a payment gateway or a secure Web server. The WebSphere Commerce Payments records its transactions in a relational database.

All integrations of the WebSphere Commerce Payments will issue order creation calls, as dictated by the underlying payment cassette. For many merchant systems that will suffice, and all other tasks will be done through the WebSphere Commerce Payments interface. Merchants who want a tighter integration of additional WebSphere Commerce Payments financial commands with other existing business formats, will want to issue additional commands, like Approve, Deposit and BatchClose.

Understanding WebSphere Commerce Payments terms

The following terms and concepts are used throughout this book:

Batch Collection of payments and credits which are settled together.

Buyer A person making an Internet purchase from the merchant.

Cashier
A component that allows merchant software to fully utilize new cassettes without requiring code modification. The cashier uses payment option profiles for each cassette to describe the required cassette-specific parameters as well as the methods of collecting that information from the merchant software environment.

Cassette
A software package that plugs into the WebSphere Commerce Payments Framework and provides support for a specific electronic payment system. Cassettes can be developed both by IBM and by third parties. Examples include the IBM SET payment protocol cassette, which supports the Secure Electronic Transaction protocol.

Credit A credit represents an interaction between a merchant and a bank when the merchant instructs the bank to refund money to the buyer.

Event listener
A registrant with the WebSphere Commerce Payments that wants to be notified when significant events occur and object states change.

Framework
The portion of the WebSphere Commerce Payments that enables different merchant servers using different payment systems to issue the same generic commands to the WebSphere Commerce Payments and use the same generic data. The WebSphere Commerce Payments uses protocol-specific cassettes to translate the generic calls to protocol-specific messages.

Merchant
A business with an Internet shopping presence. The merchant will integrate the WebSphere Commerce Payments with its merchant software.

Merchant software
The software that supports the merchant Internet business using the WebSphere Commerce Payments to process and manage Internet payments. In addition to the WebSphere Commerce Payments, this software will generally include Web-based software for browsing catalogs, managing shopping carts and placing orders. Depending on the integration level with the merchant's business, support for inventory management, shipping, and accounting software might also be included.

Order A WebSphere Commerce Payments order is an authorization from a buyer to make one or more payments using a single payment method.

Payment
A payment represents one interaction between a merchant and a financial institution to approve and capture all or part of an order. Money moves from buyer to the merchant.

Realm A registry of users along with a single method of authenticating those users. A user must be defined in a realm before being granted access to resources.

What's new for release 3.1

All cassettes (IBM provided or third party) previously installed on WebSphere Commerce Payments, Version 2.1 or higher should continue to function after successfully installing WebSphere Commerce Payments, Version 3.1.

New instructions for migrating configuration and transaction data to the latest version of the WebSphere Commerce Payments have been added. Before you install WebSphere Commerce Payments however, see the *IBM WebSphere Commerce Payments for Multiplatforms Installation Guide*

Cashier
A few important changes have been implemented for WebSphere Commerce Payments version 3.1. The principal change is that the Cashier can now pass multiple values to WebSphere Commerce Payments for each parameter. This support is particularly useful for sending order line item detail information to WebSphere Commerce Payments. It is now also possible to issue other API request via the new `issueCommand()` method. In addition, new attributes have been added to the Cashier profile Parameter element to allow null values to be passed to WebSphere Commerce Payments and also to allow sensitive data to be hidden when tracing is turned on.

User supplied order information
Allows merchant applications to specify their own transaction identifier when creating or querying WebSphere Commerce Payments orders. The intent of the transaction identifier is to allow orders to be tagged with a identifier so that the order number does not have to be used to correlate to a merchant's order or orders. Also added is the capability for the merchant to associate other user defined data with a WebSphere Commerce Payments order. AcceptPayment, ReceivePayment and QueryOrders APIs are affected by this new function.

STOP API
To increase the security of the product, the STOP API has been removed from the API set. To stop WebSphere Commerce Payments version 3.1, you should either issue the StopIBMPayServer command (ENDPYMMGR

on iSeries) from a command line or use the WebSphere Application Server adminstrative console to stop the WebSphere Commerce Payments application server.

Realm changes

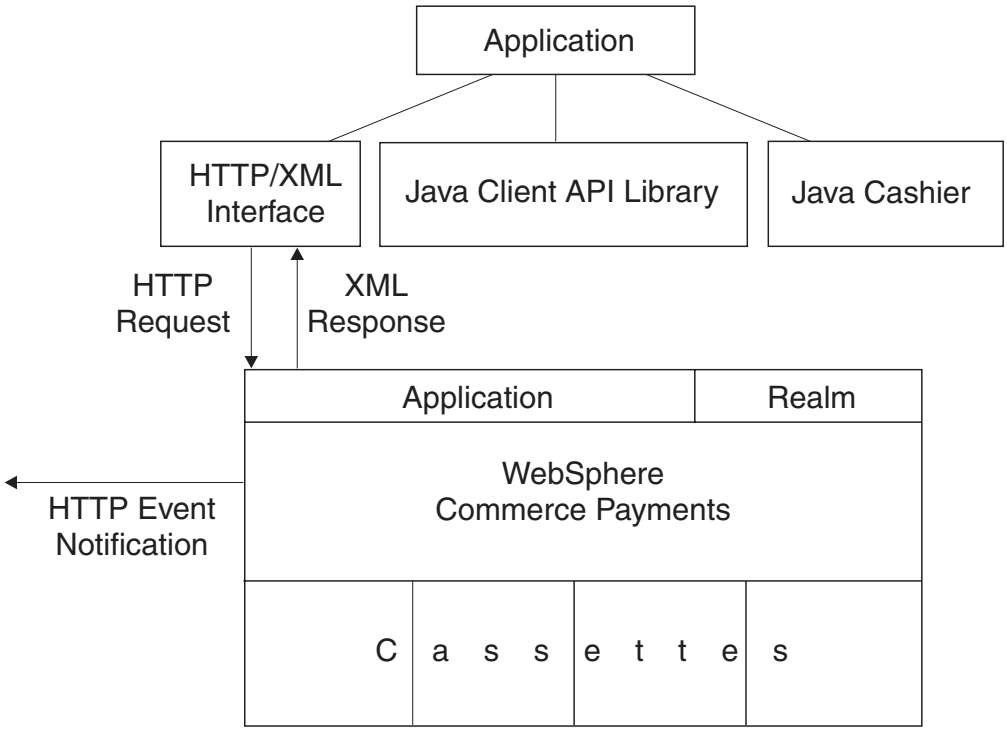
Although version 2 WebSphere Commerce Payments realms will still work in version 3.1, they must now be configured differently due to the removal of `PaymentServlet.properties` from the product. Instead of specifying the realm properties inside the PaymentServlet.properties file, they should now be defined inside a properties file unique to the realm. You must also now use the WebSphere Application Server administrative console to change the realm that WebSphere Commerce Payments uses.

Approval Expiration

An ApprovalExpiration parameter has been added to the CreateAccount and ModifyAccount commands to specify the period of time after which a given approval should be considered expired. A payment in the Approved state will enter the new ApprovalExpired state after the specificed period of time has elapsed. You can use the ApproveReversal command either to put a Payment back in the Approved state or to void the Payment. You should check your cassette documentation to determine whether Approval Expiration is supported (although note that all IBM cassettes support Approval Expiration).

Part 2. Programmer's Guide

The WebSphere Commerce Payments provides a number of programming interfaces to allow you to integrate the product into your system. The following diagram identifies these interfaces.



The central concept of the WebSphere Commerce Payments is to provide a framework for managing multiple payment systems while presenting a single interface to users. WebSphere Commerce Payments introduces the notion of a payment cassette which is a piece of plug-in software that supports a single payment system. WebSphere Commerce Payments will route incoming payment requests to the relevant cassette and responses will be as payment system-neutral as possible, thereby enabling new cassettes to be added to the system with little or no disruption to existing integration software. This publication addresses the programming interfaces that can be used by applications to integrate with WebSphere Commerce Payments and its cassettes. The cassette programming interface, which enables software developers to write new cassettes for WebSphere Commerce Payments, is fully described in the IBM WebSphere Commerce Payments Cassette Kit information at **http://www.ibm.com/software/webservers/commerce/payment/download.html**.

The main programming interface to WebSphere Commerce Payments is based on HTTP and XML standards. WebSphere Commerce Payments accepts commands as HTTP POST requests and returns XML documents embedded in HTTP responses. There are commands for all the payment processing functions and almost all of the administrative functions. Because the interface uses HTTP and XML standards, it is possible to invoke WebSphere Commerce Payments commands from a variety of programming languages. Chapter 2, "WebSphere Commerce Payments commands" on page 9 describes how these requests and responses should be formed and lists the full set of WebSphere Commerce Payments commands along with their required and optional parameters. It is

important to note that you will also need to refer to the supplemental documentation for each cassette you are using to understand the additional keywords that can be specified for each request as well as the additional cassette XML data that is provided for each response.

A Java Client API Library (CAL) is provided with WebSphere Commerce Payments that makes it easy to integrate Java software with WebSphere Commerce Payments. Using CAL, you can build Java requests and process Java response objects. CAL handles the building of the HTTP request and the parsing of the XML responses under the covers. The Client API Library is discussed in Chapter 4, "Client API Library (CAL)" on page 37.

When creating WebSphere Commerce Payments orders, it is necessary to pass information that is specific to the payment cassette that will process payments for that order. Examples of cassette-specific data include credit card numbers, check numbers, voucher IDs and expiry dates. If the code you write to handle order creation is hard-coded to support only certain cassettes, then when you add a new cassette to your system, you need to recode. To avoid this problem, the WebSphere Commerce Payments provides a Java based library of functions called the Cashier. The Cashier uses profiles - XML documents - to describe all the parameters that are required by a cassette for order creation. That way, if you use the Cashier to create WebSphere Commerce Payments orders, you will not need to write order creation code that is specific to any given cassettes. The Cashier is described in Chapter 3, "Cashier" on page 19.

Every WebSphere Commerce Payments command must identify the user that is issuing the command. WebSphere Commerce Payments will ensure that the user's credentials are valid - usually by checking a password - and that the user has permission to perform the command. To support these security checks, WebSphere Commerce Payments needs access to a list of users and it needs to know how it can authenticate any given user. This mechanism is known as a *realm* and, although WebSphere Commerce Payments provides a simple realm that allows you to administer a list of user and their passwords, it is possible for you to write a WebSphere Commerce Payments realm that integrates with your existing systems. Writing WebSphere Commerce Payments realms is discussed in Chapter 6, "WebSphere Commerce Payments realm support" on page 49.

WebSphere Commerce Payments provides an event notification mechanism that can alert you when certain events occur. The supported event triggers include the starting or stopping of the WebSphere Commerce Payments and its cassettes, the change in status of orders belonging to a given merchant or special events defined by particular cassettes. You can tell WebSphere Commerce Payments which events you are interested in. When the event is triggered, WebSphere Commerce Payments will create an HTTP POST message and send it to the URL you specified. You will need to write a servlet or CGI program - known as an event listener - to process event notifications. The event notification mechanism is described in Chapter 5, "Event notification" on page 45.

Chapter 2. WebSphere Commerce Payments commands

Merchant business software can issue administration, payment and query commands to the WebSphere Commerce Payments. These commands consist of keyword-value pairs. (See Chapter 7, “WebSphere Commerce Payments command reference” on page 59, for command tables.) Commands are executed by issuing requests and waiting for the responses. WebSphere Commerce Payments requests are formatted as HTTP POST messages. WebSphere Commerce Payments responses are XML documents embedded in HTTP. (For a detailed description of the XML objects, and associated fields, see Chapter 8, “WebSphere Commerce Payments data” on page 111.) This chapter describes the HTTP POST, communication with the WebSphere Commerce Payments and the XML output.

WebSphere Commerce Payments requests

Merchant software issues commands to the WebSphere Commerce Payments by creating an HTTP POST message and sending it to the WebSphere Commerce Payments. Like any HTTP POST message, a command consists of a header and a body. Following is an example of a WebSphere Commerce Payments command:

```
POST /webapp/PaymentManager/PaymentServlet HTTP/1.1
Connection: Keep-Alive
Accept: application/xml
PM-Accept-Language: en-US
Authorization: Basic YWRtaW46YWRtaW4=
Host: localhost
User-Agent: Java PaymentServerClient
Content-Encoding: 8859_1
Content-Length: 187
Content-Type: application/x-www-form-urlencoded

OPERATION=ACCEPTPAYMENT&ETAPIVERSION=3&PAYMENTTYPE=OfflineCard
&MERCHANTNUMBER=123456789&ORDERNUMBER=91600886&AMOUNT=500
&CURRENCY=840&%24EXPIRY=200212&%24PAN=5015550000033019&%24BRAND=ROBO
```

Note: The breaks in the HTTP body in the example above are for formatting purposes only. Syntax has to be on the same line.

WebSphere Commerce Payments commands require the header to contain a number of specified keyword-value pairs, encoded in a particular format. The HTTP header must contain the following fields with these values:

```
POST /webapp/PaymentManager/PaymentServlet HTTP/1.1
Connection: Keep-Alive
Accept: application/xml
Content-Encoding: 8859_1
Content-Type: application/x-www-form-urlencoded
```

In addition, the header must contain additional fields with calculated values:

Host: <PaymentServer host>
This should be the TCP/IP hostname of WebSphere Commerce Payments

Content-Length: <length>
The length of the HTTP body in bytes

Authorization: Basic <authorization-string>
The authorization string consists of a userid and password string, separated by a single colon (″:″) character. The string should be encoded with a base64 encoding.

```
authorization-string=base64–user-pass
base64-user-pass=<base64encoding of user-password,
except not limited to 76 char/line>
user-pass=user":"password
userid=*<TEXT excluding ":">
password=*TEXT
```

Optionally, the HTTP header may contain a PM-Accept-Language HTTP header. This header indicates the language in which WebSphere Commerce Payments should provide return code messages in the response message.
PM-Accept-Language: Locales should be specified according to the HTTP RFC 2068. Locales supported by WebSphere Commerce Payments include: **pt** (Brazilian Portuguese), **en** (English), **fr** (French), **de** (German), **it** (Italian), **ja** (Japanese), **ko** (Korean), **zh** (simplified Chinese), **es** (Spanish), **zh_TW** (traditional Chinese). Note that, although more than one locale can be specified on the PM-Accept-Language HTTP header, WebSphere Commerce Payments will only use the first locale. If no PM-Accept-Language header is sent, WebSphere Commerce Payments will use the locale of the machine where WebSphere Commerce Payments is installed.

The client or merchant programmer may wish to specify additional header fields, to use HTTP functionality beyond the minimal WebSphere Commerce Payments communication requirements. The interpretation of these fields is dependent on the network environment and the Web server under which the WebSphere Commerce Payments is installed.

The HTTP body

The body of a WebSphere Commerce Payments command consists of a set of keyword-value pairs, formatted using the encoding specified by the HTTP content-type: application/x-www-form-urlencoded.

Keywords can be included multiple times (for example, multiple order numbers specified in a query order command).

The command body must be formatted according to the following rules:

- Each WebSphere Commerce Payments command parameter and its associated argument (each keyword-value pair of a WebSphere Commerce Payments command), are separated from each other by an equals (″=″) character.
- Each keyword-value pair is separated from other keyword-value pairs by an ampersand (″&″) character.
- The keywords and values are URL encoded, which is also the way that binary data is sent to the WebSphere Commerce Payments. Rules for URL encoding follow:
  - All space characters (hex 0X20 ASCII characters) are replaced by ″+″ characters (hex 0x2B characters)
  - All bytes of each keyword and value that do not map to an alphanumeric US-ASCII character must be escaped. Each of these bytes are replaced with the escape sequence ″%HH″ where HH is the two hexadecimal digits representing the ASCII code of the character (byte).
- Keywords are case insensitive. Values are case sensitive

Character set issues

All WebSphere Commerce Payments keywords are specified in the US-ASCII character set. Values must be encoded in the UTF-8 character set prior to the

URL-encoding of the HTTP POST body. For example, the Unicode character 0x3053 is represented in UTF-8 as 0xE3, 0x81, 0x93. Once this value is URL encoded, it is %E3%81%93.

Note: For US-ASCII string or numeric values, no translation is necessary.

Although WebSphere Commerce Payments provides its own authentication mechanisms, it is possible to plug in your own custom realm code and have WebSphere Commerce Payments use that (see Chapter 6, “WebSphere Commerce Payments realm support” on page 49). If the realm that you are using uses userid and password, then you should use the Authorization HTTP header to encode these in the request message. However, if your realm uses other authentication data, then this should be sent to WebSphere Commerce Payments via the PMAUTHOBJECT keyword-value pair in the HTTP body.

Communication

To send a command to the WebSphere Commerce Payments:

1. Open a TCP connection to the WebSphere Commerce Payments host and port. The port is usually 80, unless reconfigured.
2. Send a request and wait for the response.
3. Close the connection.

If communication fails prior to receipt of the response, it is uncertain whether or not the WebSphere Commerce Payments command has actually executed. To determine if the command has executed, issue query commands to confirm that the command was received and processed..

If you want to use SSL, configure the Web server on the WebSphere Commerce Payments to support SSL connections. Once the Web server is configured for SSL, you can send commands using SSL. You must be ready to participate and perform all steps to create SSL communication.

WebSphere Commerce Payments responses

WebSphere Commerce Payments responses are XML documents, embedded in HTTP. The format of the XML document is defined in the WebSphere Commerce Payments Document Type Definition (DTD). *IBMPaymentServer.dtd* contains the DTD and this file can be found in the */include* subdirectory.

For **iSeries**, this file can be found in the */QIBM/ProdData/PymSvr/XML/DTD* subdirectory.

Every HTTP response contains an XML document with a PSApiResult element that identifies the primary and secondary return code, along with an object count and additional return code messages, which may contain descriptions of any WebSphere Commerce Payments return code pairs. For a description of primary and secondary return code values, see Appendix A, “WebSphere Commerce Payments return codes” on page 129.

Cassette specific objects are represented using the cassette object and cassette configuration elements. Details about individual properties can be found in the respective cassette supplement. (See the cassette supplement documentation for more information)

Query commands will additionally contain descriptions of WebSphere Commerce Payments objects and the number of objects returned. Framework objects are described in the DTD (Document Type Definition) and in the object definition tables found in Chapter 8, "WebSphere Commerce Payments data" on page 111. The DTD for this XML document can be either:

- Included in the response
- Found in the file *IBMPaymentServer.dtd*

When the WebSphere Commerce Payments successfully receives, processes and responds to a request, it returns an HTTP status code of 200. Other HTTP status codes can be returned by the web server, due to situations like an authentication failure or when WebSphere Application Server is not running. This status code, along with any information in the body, indicates the source of the problem.

Formatting commands

Following are two examples of XML documents resulting from an AcceptPayment command, and a QueryOrders with Payments command.

AcceptPayment

```
POST /webapp/PaymentManager/PaymentServlet HTTP/1.1
Connection: Keep-Alive
Accept: application/xml
PM-Accept-Language: en-US
Authorization: Basic YWRtaW46YWRtaW4=
Host: localhost
User-Agent: Java PaymentServerClient
Content-Encoding: 8859_1
Content-Length: 187
Content-Type: application/x-www-form-urlencoded

OPERATION=ACCEPTPAYMENT&ETAVERSION=3&PAYMENTTYPE=OfflineCard&MERCHANTNUMBER=
123456789&ORDERNUMBER=94184938&AMOUNT=500&CURRENCY=840&%24EXPIRY=
200212&%24PAN=5015550000033019&%24BRAND=ROBO
-------------------------------------
<?xml version="1.0" encoding="UTF-8"?>
<PSApiResult objectCount="0" primaryRC="0" secondaryRC="0">
</PSApiResult>
```

QueryOrders with Payments

The following example is a response to a QueryOrder with Payment command. There are two order objects contained in the response document:

- First order object contains one payment
- The second order object does not contain any payments

```
POST /webapp/PaymentManager/PaymentServlet HTTP/1.1
Connection: Keep-Alive
Accept: application/xml
PM-Accept-Language: en-US
Authorization: Basic YWRtaW46YWRtaW4=
Host: localhost
User-Agent: Java PaymentServerClient
Content-Encoding: 8859_1
Content-Length: 100
Content-Type: application/x-www-form-urlencoded

OPERATION=QUERYORDERS&ETAVERSION=3&PAYMENTTYPE=OfflineCard&MERCHANTNUMBER=
123456789&WITHPAYMENTS=1
-------------------------------------
<?xml version="1.0" encoding="UTF-8"?>
<PSApiResult objectCount="2" primaryRC="0" secondaryRC="0">
  <OrderCollection size="2" withCredits="0" withPayments="1">
    <PSOrder ID="O:123456789:94184938" amount="500" amountExp10="-2"
```

```
approvesAllowed="1" brand="ROBO" currency="840" merchantAccount="1"
merchantNumber="123456789" merchantOriginated="1" numberOfCredits="0"
numberOfPayments="1" orderNumber="94184938" paymentType="OfflineCard"
state="order_refundable" timeStampCreated="966461827000"
timeStampModified="966463091000" unapprovedAmount="0">
      <PaymentCollection size="1" withOrders="0">
        <PSPayment ID="P:123456789:94184938:1" amountExp10="-2"
approveAmount="500" currency="840" depositAmount="0" merchantAccount="1"
merchantNumber="123456789" orderNumber="94184938" paymentNumber="1"
paymentType="OfflineCard" state="payment_approved"
timeStampCreated="966463091000" timeStampModified="966463092000">
          <CassetteExtensionObject>
          </CassetteExtensionObject>
        </PSPayment>
      </PaymentCollection>
      <CassetteExtensionObject>
        <CassetteProperty propertyId="Expiry" value="200212">
        </CassetteProperty>
        <CassetteProperty propertyId="AccountNumber" value="1">
        </CassetteProperty>
        <CassetteProperty propertyId="Brand" value="ROBO">
        </CassetteProperty>
        <CassetteProperty propertyId="AmountApproved" value="500">
        </CassetteProperty>
        <CassetteProperty propertyId="Pan" value="5015550000033019">
        </CassetteProperty>
      </CassetteExtensionObject>
    </PSOrder>
    <PSOrder ID="O:123456789:92005267" amount="500" amountExp10="-2"
approvesAllowed="1" brand="ROBO" currency="840" merchantAccount="1"
merchantNumber="123456789" merchantOriginated="1" numberOfCredits="0"
numberOfPayments="0" orderNumber="92005267" paymentType="OfflineCard"
state="order_refundable" timeStampCreated="966459650000"
timeStampModified="966459650000" unapprovedAmount="500">
       <PaymentCollection size="0" withOrders="0">
       </PaymentCollection>
       <CassetteExtensionObject>
         <CassetteProperty propertyId="Expiry" value="200212">
         </CassetteProperty>
         <CassetteProperty propertyId="AccountNumber" value="1">
         </CassetteProperty>
         <CassetteProperty propertyId="Brand" value="ROBO">
         </CassetteProperty>
         <CassetteProperty propertyId="AmountApproved" value="0">
         </CassetteProperty>
         <CassetteProperty propertyId="Pan" value="5015550000033019">
         </CassetteProperty>
       </CassetteExtensionObject>
     </PSOrder>
   </OrderCollection>
  </PSApiResult>
```

WebSphere Commerce Payments command security

When WebSphere Commerce Payments receives a command issued by the user, it will process the command as follows:

- Authenticate the user by the realm.
- Authorizes the user through the access control facility.
- Process the command.

The following sections describe the concepts associated with command security.

Users

WebSphere Commerce Payments user information is captured in two places:

1. in a realm
2. in the WebSphere Commerce Payments database

WebSphere Commerce Payments authenticates users through the use of realms. A realm is a registry of users that is responsible for managing the user's name, password, and perhaps some other form of user identification (for more information on realms, see Chapter 6, "WebSphere Commerce Payments realm support" on page 49). A WebSphere Commerce Payments user must be defined in a realm before being granted access to WebSphere Commerce Payments resources. Payments and Merchant administrators can use the WebSphere Commerce Payments API command or the WebSphere Commerce Payments user interface User window to assign access to a user defined in a realm.

Authentication

WebSphere Commerce Payments supports two methods to authenticate users:

1. Basic authentication (through the use of realms)
2. WebSphere Commerce Payments Authentication Object (PMAUTHOBJECT)

Both of these methods are described in detail in "Authentication mechanism" on page 51.

Role-based access control

WebSphere Commerce Payments employs a role-based access control scheme which defines four WebSphere Commerce Payments roles:

1. Payments administrator
2. Merchant administrator
3. supervisor
4. clerk

The user's role determines which commands can be issued by that user.

A user other than the Payments administrator can associate with multiple merchants. For example, a merchant administrator can manage more than one merchant. Similarly, supervisors and clerks can issue commands for multiple merchants. The WebSphere Commerce Payments supports the following role-based access scenarios:

- Payments administrators can issue all of the API commands for all merchants.
- Merchant administrators can perform all functions for the merchants with whom they associate, except for several limitations on the SetUserAccessRights and the QueryUsers commands (for more information on these commands, see "QueryUsers" on page 102 and "SetUserAccessRights" on page 108).
- Supervisors and clerks can issue limited commands for the merchants with whom they associate.

Assigning a user's access permissions

A user's permission (or role) can be assigned or changed only by the Payments administrator or the Merchant administrator. The Payments administrator can assign or change *any* user's access rights and can assign or change a user's role to whatever he wants that user's role to be, including the role of Payments administrator.

The Merchant administrator can only assign or remove a user as a Merchant administrator, supervisor, or clerk. Further, the merchant administrator can do so only under one of the following conditions:

1. If the user being granted access to multiple merchants does not currently have access rights in the WebSphere Commerce Payments, then the merchant administrator can grant this user access *only* to merchants that he (as the merchant administrator) already has access to.
2. If the user being granted access to multiple merchants *does* have access rights in the WebSphere Commerce Payments, then the merchants with whom he is currently associated should also be associated with the assigning merchant administrator. Further, the merchants who are being assigned to associate with the user should also be a subset of the merchants associated with the assigning merchant administrator.

For example, the user `X` is the merchant administrator for merchants `A`, `B`, and `C`. User `Y` does not have access rights in the WebSphere Commerce Payments. `X` can assign `Y` as the merchant administrator for the merchants `A`, `B`, and `C` or for the merchants `A` and `B`. However, if the user `Y` has access rights in merchants other than `A`, `B` and `C` (for example the user `Y` is the merchant administrator for the merchant `D`), then the user `X` cannot change the user `Y`'s access rights.

The following figure uses set notation to represent some typical scenarios.

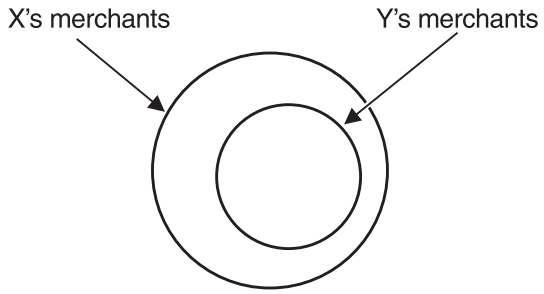


Figure A

In figure A, user `X` has given access rights to some of `X`'s merchants to user `Y`.

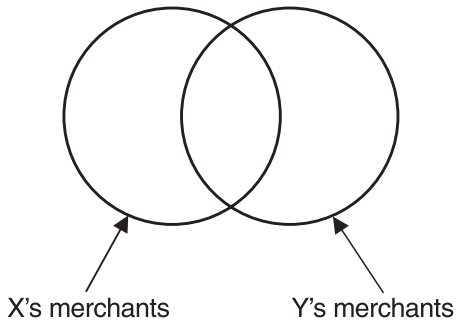


Figure B

In figure B,X and Y are associated with different sets of merchants even though they both associate with some common merchants. In this case, neither X nor Y can change the other's permissions even though they are both merchant administrators for a common set of merchants.

Role permissions table

Each role has an associated set of operations that can be performed by a user having this role. The following notation is used to describe the capabilities of each role listed in Table 2 below.

Table 2. Field Values for Role capabilities

Field value	Role capabilities
Y	Allowed to perform the command.
M	Allowed to perform the command, if the user's merchant number includes all the merchant numbers in the command. If there is no merchant number specified in the command, authorization fails.
u	Allowed to perform the command, if the user attempting the command matches the user specified in the command. If there is no user specified in the command, authorization fails.
a	Allowed to perform the command, if SETUserAccessRights special authorization logic allows it. For more information on the SETUserAccessRights command, see "SetUserAccessRights" on page 108.
m	Allowed to perform the command, if QueryUsers special authorization logic allows it. For more information on the QueryUsers command, see "QueryUsers" on page 102.
<blank>	A user with this role is not allowed to perform the command.

The following table illustrates the capabilities each role has. An asterisk (*) following a command indicates that the command does not have a required merchant number parameter.

Table 3. Role capabilities

Command	Payment Server Admin	Merchant Admin	Supervisor	Clerk
About*	Y	Y	Y	Y
AcceptPayment	Y	M	M	M
Approve	Y	M	M	M
ApproveReversal	Y	M	M	M
BatchClose	Y	M	M	M
BatchOpen	Y	M	M	M
BatchPurge	Y	M	M	M
CancelOrder	Y	M	M	
CassetteControl	Y	M	M	M
CloseOrder	Y	M	M	
CreateAccount	Y	M		
CreateMerchant	Y			
CreateMerchantCassetteObject*	Y	M		

Table 3. Role capabilities (continued)

CreateMerEventListener	Y	M		
CreatePaySystem	Y			
CreateSNMEventListener	Y			
CreateSystemCassetteObject*	Y			
DeleteAccount	Y	M		
DeleteBatch	Y	M	M	M
DeleteMerchant	Y			
DeleteMerchantCassetteObject*	Y	M		
DeleteMerEventListener	Y	M		
DeletePaySystem	Y			
DeleteSNMEventListener	Y			
DeleteSystemCassetteObject*	Y			
Deposit	Y	M	M	M
DepositReversal	Y	M	M	
ModifyAccount	Y	M		
ModifyCassette*	Y			
ModifyMerchant	Y	M		
ModifyMerchantCassetteObject*	Y	M		
ModifyMerEventListener	Y	M		
ModifyPayServer*	Y			
ModifyPaySystem	Y			
ModifySNMEventListener	Y			
ModifySystemCassetteObject*	Y			
ModifyUserStatus	Y	M		
QueryAccounts	Y	M	M	M
QueryBatches	Y	M	M	M
QueryCassettes	Y			
QueryCredits	Y	M	M	M
QueryEventListeners	Y	M		
QueryMerchants	Y	M	M	M
QueryOrders	Y	M	M	M
QueryPayments	Y	M	M	M
QueryPaymentServer	Y			
QueryPaySystems	Y	M	M	M
QueryUsers	Y	m	u	u
ReceivePayment	Y	M	M	M
Refund	Y	M	M	
RefundReversal	Y	M	M	
SetUserAccessRights*	Y	a		

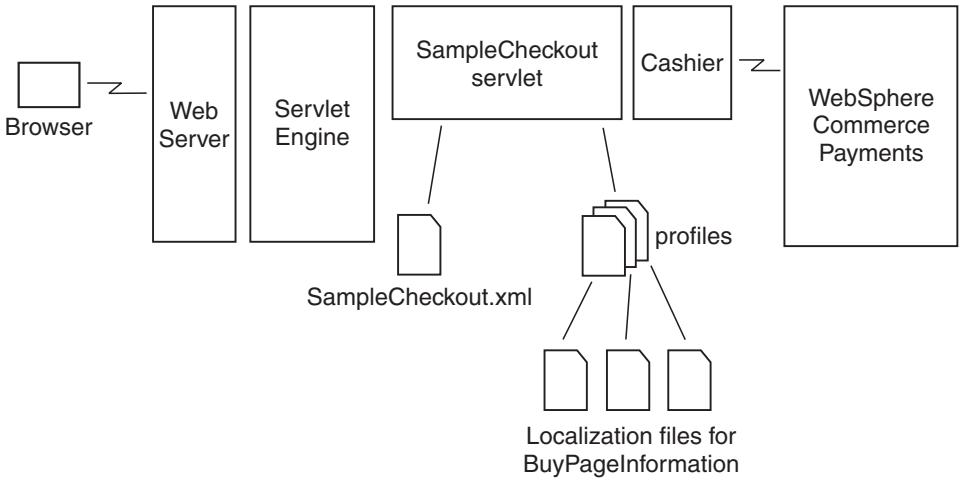
Note: A user may not update himself. That is to say, user ″admin″ may not call SETUSERACCESSRIGHTS with the user parameter set to ″admin″.

Chapter 3. Cashier

Introduction to the Cashier

The Cashier is WebSphere Commerce Payments code that can be invoked by client applications - such as merchant software - to simplify the process of creating WebSphere Commerce Payments orders and other WebSphere Commerce Payments commands. The Cashier uses XML documents called profiles that describe how commands such as orders should be created for a given cassette. This allows the client code writer to concentrate on integrating with the WebSphere Commerce Payments in a generic way rather than having to write code that deals with cassette-specific information.

You can still create WebSphere Commerce Payments orders without using the Cashier; programs can use the AcceptPayment and ReceivePayment APIs. However, the use of the Cashier is preferred since it allows the potential for new cassettes to be introduced to the system without the need for rewriting any code.



The Cashier is written in Java and is included in the eTillCal.zip package. It provides a set of methods that can be invoked directly by a WebSphere Commerce Payments client application. The Cashier itself uses the Client API Library (CAL) to send AcceptPayment, ReceivePayment, and other commands to WebSphere Commerce Payments. Therefore, the Cashier benefits from all the advantages of CAL. The code can operate remotely from WebSphere Commerce Payments, and can be configured to use a socks server and encrypt messages via SSL if required. The profiles used by the Cashier must be available where the Cashier is running.

The principal Cashier method is `collectPayment()`; this is the method that client applications must invoke to create a WebSphere Commerce Payments order. `collectPayment()` takes a profile name, the locale, and a list of environment variables as arguments. It loads the corresponding profile and uses it to build an AcceptPayment or ReceivePayment API request. The environment variables are used to supply the API parameter values.

To query the state of the WebSphere Commerce Payments orders and payments, you can use the `checkPayment()` method at any time after the `collectPayment()` call.

Additionally, you can use the `issueCommand()` to build and issue other API requests on WebSphere Commerce Payments. Currently the only supported API through `issueCommand()` is the Deposit API.

Cashier profiles

Cashier profiles are XML documents that describe how WebSphere Commerce Payments commands should be created. All profiles must include the following:

- required WebSphere Commerce Payments parameters
- required cassette parameters
- specifications for how the Cashier supplies values for the above parameters

Profiles may include the following:

- an indication of whether a wallet is used - this flag determines whether the Cashier will issue an AcceptPayment command or a ReceivePayment command.
- indication of which WebSphere Commerce Payments instance to use for each profile
- optional WebSphere Commerce Payments parameters
- optional cassette parameters
- buy page information that specifies how client code should build buy pages to collect buyer information. For example, an HTML form that collects credit card information required by a specific cassette
- an indication of whether diagnostic information is to be enabled for the profile

Cashier profiles allow parameter values to be specified in four ways:

1. hard-coded as constants in the profile
2. passed as an environment variable on the `collectPayment()` or `issueCommand()` calls
3. specified as originating from a relational database field
4. specified as being calculated by Cashier extension code

If your system already has easy access to data needed by your profiles, then it is practical to pass this data to the Cashier in environment variables on the `collectPayment()` or `issueCommand()` calls. If data is difficult for your system to obtain, or is required by only a few profiles, then passing this as environment variables may be inefficient because you will be deriving this data for all calls to `collectPayment()` or `issueCommand()`, whether the data is required or not.

Designing your integration

This section describes considerations for writing code that integrates with the WebSphere Commerce Payments via the Cashier.

Managing Cashier profiles

Before using the Cashier, you should determine which profiles you want to make available on your site. Cassette writers should provide Cashier profiles that are adaptable for your use. These profiles are stored in the profiles subdirectory of WebSphere Commerce Payments. However, even if a cassette does not provide any profiles, they can easily be created by following the cassette's supplement guide and the instructions in "Writing cashier profiles" on page 23.

If your system supports multiple stores or merchants, you must decide how you will determine which profiles are in active use. In the simplest case, all merchants or

stores may always use a single set of profiles. However, in a more complex scenario, different merchants or stores may support different sets of profiles. In this case, you must provide support to map these merchants or stores to the profiles they use. You may also need to provide tools to administer this table of merchant to profile mappings.

Mapping merchant numbers

Merchants are the objects with which cassettes are associated and against which orders are placed. They are identified by merchant numbers of up to nine numeric digits. Because you can create more than one merchant number for each merchant entity in your system, it is important to consider how to map the merchant or store entities in your system against merchant numbers in WebSphere Commerce Payments. If there is a one-to-one correspondence, and the identifier you use for your merchant or store can be represented as a string of up to nine digits, then you need not store WebSphere Commerce Payments merchant numbers in your system. Otherwise, you must decide how to store WebSphere Commerce Payments merchant numbers as foreign keys.

Mapping order numbers

Orders are identified by order numbers of up to nine digits. Each order number must be unique for each merchant number, so it is theoretically possible for a single instance of WebSphere Commerce Payments to have 999,999,999 merchants, each with 999,999,999 orders. (Of course, practical limitation would become unmanageable before reaching these limits.)

A WebSphere Commerce Payments order has a specific definition that might not precisely match the use of an order in your system. Each WebSphere Commerce Payments order is an authorization from a buyer to make one or more payments against a particular payment instrument. An order in which a buyer uses multiple payment methods must be represented in WebSphere Commerce Payments as multiple orders. An example would be a down-payment paid by credit card with the balance paid later by check.

You decide how to map orders in your system with orders in WebSphere Commerce Payments. If necessary, you may need to store one or more WebSphere Commerce Payments order numbers with each order in your system.

Designing profiles

Because each profile contains data specifying how to derive the values of parameters for WebSphere Commerce Payments and cassettes, profiles are usually specific to a single integration and cannot be copied to another system without modification. This section lists some of the things to consider when designing how profiles will work for your integration.

WebSphere Commerce Payments Configuration

There are two ways to configure your system to point to one or more WebSphere Commerce Payments instances:

1. by specifying the WebSphere Commerce Payments configuration inside each Cashier profile
2. by specifying the configuration inside your application and using that configuration for all Cashier profiles.

Either way, it is important to understand that an order is managed by a single WebSphere Commerce Payments instance. Therefore, when the order is created, you must record which WebSphere Commerce Payments instance owns the order.

If you use profile-based configuration, you can do this by storing the profile name along with the order. Later, when you want to perform payment operations on the order, you can query the Cashier to discover which WebSphere Commerce Payments instance owns the order and direct your API requests to that instance.

Profile parameter sources

Remember to keep in mind that either `collectPayment()` or `issueCommand()` can be used, but that we recommend going to the use of `issueCommand()`.

When using the Cashier, you must decide where your profiles will get their API parameter values.

If your system already has easy access to data needed by your profiles, then it is practical to pass this data to the Cashier in environment variables on the `collectPayment()` `issueCommand()` call. If data is difficult for your system to obtain, or is required by only a few profiles, then passing this as environment variables may be inefficient because you will be deriving this data for all calls to `collectPayment()` `issueCommand()`, whether the data is required or not.

In these cases, you may prefer to have the Cashier derive the data itself. If the data is available in a relational database, you can code your profiles to instruct the Cashier to perform a database query to get it. Or, you can write Cashier extension code to derive the parameter value. Refer to “Writing your integration” on page 27 to see how this can be done.

Buy page information

Using the Cashier and profiles allows your WebSphere Commerce Payments integration to support the addition of future payment cassettes without the need for recoding your system. New cassettes will require different payment information to be collected on the buy page. Even within the set of credit card cassettes, there are differences in the buy pages that are presented to a buyer. For example, some cassettes support the *Address Verification Service* (AVS) and others do not.

If you write your integration to use information in the Cashier profiles to build buy pages it becomes much easier to support new cassettes by avoiding the need to recode your buy pages for the new cassettes.

The profile’s buy page information is determined entirely by your integration design. It could contain the HTML required to build a form to present to the buyer; it could be an XML document that describes the data that should be collected; or it could be a pointer to a Java Server Page or Active Server Page that collects the data. The only thing you must ensure is that the data entered by the buyer is made available when the Cashier is using the profile’s parameter definitions to build the WebSphere Commerce Payments API request.

Publish profile interface

One of the major advantages of the Cashier is that other people can write profiles that work with your integration. Having integrated with WebSphere Commerce Payments using the Cashier, new cassettes can be supported by providing the relevant Cashier profiles, with no requirement for program code changes. To publish the interface, include the specification for buy page information, parameter sources, and whether profiles need to contain WebSphere Commerce Payments configuration information.

AVS

WebSphere Commerce Paymentscassettes return AVS result codes to merchants on financial transactions. Because these codes are cassette-specific (meaning they vary by WebSphere Commerce Paymentscassette), WebSphere Commerce Paymentsprovides a set of common AVS result codes to extend the cassette-specific codes. For a mapping of the common AVS result codes to the cassette-specific codes, see “AVS common codes” on page 117.

Trace

The Cashier provides a trace mechanism that allows diagnostic information to be written directly to your own system logs, simplifying the process of diagnosing problems. This facility writes all trace information to one log, thus avoiding the difficulties involved with correlating multiple logs. To use this facility, follow the instructions in “Writing your integration” on page 27. If integrating this trace information is not required for your system, the Cashier provides a simple trace class that writes the diagnostic information directly to a flat file.

Recording trace information represents a small performance overhead. For this reason, tracing can be enabled and disabled on a per profile basis. The Profile element supports an enableTrace attribute that allows you to control tracing.

Error log

Although the Cashier provides a trace facility for use by service personnel in diagnosing problems, it does not record errors for use by users. Instead, the Cashier throws Java Exceptions when an error condition is detected. It is the responsibility of your system to catch these Exceptions and report them appropriately to the user.

Writing cashier profiles

Four things are required to write a cashier profile:

1. knowledge of the structure of cashier profiles
2. specifications of both the required and optional WebSphere Commerce Payments parameters
3. specifications of both the required and optional cassette parameters
4. specification of the integration with the cashier

If the cassette writer provides cassette profiles, they are stored in the profiles directory where WebSphere Commerce Payments is installed. These profiles can easily be copied and modified to work with other systems. If no template profile is available, then you must construct a new profile.

Basic Profile Structure

Cashier profiles are XML documents that implement the profile.dtd document type definition. They have the following basic structure:

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE Profile SYSTEM "profile.dtd">
<Profile useWallet="false" enableTrace="false">

...

</Profile>
```

When the useWallet attribute is set to true, the Cashier builds a ReceivePayment API request for `collectPayment()`; when set to false, an AcceptPayment API request is constructed for collectPayment(). The enableTrace attribute indicates whether diagnostic information should be recorded when the Cashier is using this profile.

WebSphere Commerce Payments configuration in profiles

Optionally, profiles can also contain a WebSphere Commerce Payments configuration element as follows:

```
<Profile useWallet="false">

  <PaymentsConfiguration
     hostname="..."
     port="80"
     userid="admin"
     password="admin"
     useSSL="true"
     socksHostname="..."
     socksPort="..."
dtdPath="..."
   />

  ...

</Profile>
```

This information indicates how the Cashier communicates with a WebSphere Commerce Payments instance when using this profile. The hostname and port attributes identify the socket where WebSphere Commerce Payments is listening for API requests. The userid and password attributes specify the identity and credentials that the Cashier should assume when building API requests. socksHostname and socksPort are optional attributes that indicate the socks server to use, if any. useSSL is a flag that indicates whether the communication with WebSphere Commerce Payments should be encrypted using SSL. The optional dtdPath parameter specifies the path of the WebSphere Commerce Payments DTD.

Select statements

If the merchant integration supports the use of relational database queries to derive values for parameter values, then the profile may also contain one or more SelectStatement elements.

```
<Profile useWallet="false">

  <SelectStatement id="..." allowMultiples="...">
    SELECT * FROM ... WHERE ...
  </SelectStatement>

  ...

</Profile>
```

The contents of the element form the SQL query statement. The id attribute specifies an identifier for the statement that can be used in subsequent DatabaseValue elements to refer back to this statement. When building an SQL statement which does not send back repeating data, ensure that the statement returns exactly one row, which can be accomplished by not specifying the allowMultiples attribute, or specifying allowMultiples=″false″. In this case, the Cashier reports zero or more then one row as errors.

The optional allowMultiples attribute, if ″true″, indicates that the SQL query may return multiple rows of data. In this case, the cashier will create multiple arguments in the API request for each database parameter that references the select statement. There will be as many arguments as there are rows returned from the query and each argument will be distinguished by adding a period and an incrementing integer to the end of the argument. For example, if a parameter with ID $LINEITEM references a select statement with allowMultiples set to ″true″, and the SQL query returns three rows, then three arguments will be generated in the API request by the cashier: $LINEITEM.1, $LINEITEM.2 and $LINEITEM.3

CollectPayment

The CollectPayment element contains all the data needed to create WebSphere Commerce Payments orders using the Cashier.

```
<Profile useWallet="false">
  <CollectPayment>

    ...

  </CollectPayment>
</Profile>
```

Command

The Command element contains all the data needed to create WebSphere Commerce Payments commands using the Cashier. Although this command can be used to build and issue any WebSphere Commerce Payments API request, the only API currently supported is Deposit.

```
<Profile useWallet="false">
  <Command name="DEPOSIT">

    ...

  </Command>
</Profile>
```

Buy Page Information

The system that integrates with the Cashier specifies whether a BuyPageInformation element is required, and if so, what format it must take.

```
  <BuyPageInformation reference="..."

    ...

  </BuyPageInformation>
```

BuyPageInformation elements are valid within either CollectPayment or Command elements.

The optional reference attribute is a free-form field. Its use is defined by the system that integrates with the Cashier. Read the documentation to see if and how this field should be used.

Parameters

Parameter elements specify how the Cashier can derive values for each parameter on the WebSphere Commerce Payments API request.

```
<name="ACCEPTPAYMENT">

  <Parameter
```

```
name="..."
encoding="... "
maxBytes="..."
sensitive="..."
allowNullValue="...">
...
</Parameter>
```

Parameter elements are valid within either CollectPayment or Command elements.

The name attribute indicates the name of the API parameter keyword that is sent to Payment Manger. The element contents indicate how the value should be derived. There are four ways to derive these values: constants, variables, database entries, and extensions.

The optional encoding attribute is used if the parameter needs to be in a particular character encoding. The value is a valid Java name for an encoding. The default encoding is UTF8.

The optional maxBytes attribute is used to limit the number of bytes of the parameter passed to WebSphere Commerce Payments. This can be useful to prevent a parameter containing non-critical data from causing a command to fail because the parameter value is too long.

The optional sensitive attribute, when set to ″true″ ensures that the cashier will not display the value of the parameter in the cashier trace file. This is useful for protecting sensitive data, such as credit card numbers from being obtained illicitly.

Constant parameters

Constant parameters allow unchanging parameter value to be hard-coded inside the profile.

```
<Parameter name="..."><CharacterText>1</CharacterText></Parameter>
```

Variable parameters

Environment variable parameters specify that the value for the parameter is provided by the system that integrates with the Cashier. Environment variable values are specified by enclosing the variable name in curly braces {} inside the Parameter element content. The Cashier reports an error if a specified variable was not passed in on the `collectPayment()` call.

```
<Parameter name="..."><CharacterText>{var1}{var2}</CharacterText></Parameter>
```

Database parameters

Database parameters indicate that a value is derived by performing a query on a relational database and looking in the column indicated by the columnName attribute for the result. The statementID attribute refers to the id attribute of a previously declared SelectStatement element. The Cashier reports an error if the query cannot be performed or the column name does not exist.

```
<Parameter name="...">
  <DatabaseValue statementID="..." columnName="..."/>
</Parameter>
```

Extension parameters

Extension parameters indicate that a custom-written program must be executed to derive a parameter value. The `name` attribute of the `ExtensionValue` element indicates the name of the program to run. See “Writing your integration” on page 27 for details about writing Cashier extensions.

```
<Parameter name="...">
  <ExtensionValue name="..."/>
</Parameter>
```

Writing your integration

The following topics discuss the requirements for writing your integration. They are as follows:

- Building profiles
- Including the necessary files
- Creating a Cashier object
- Creating orders in the WebSphere Commerce Payments with `collectPayment()`
- Checking the status of an order with `checkPayment()`
- Using BuyPageInformation
- Tracing
- Exceptions
- Writing extensions

Note: Javadoc is provided for the Cashier in the **/WebSphere/AppServer/InstalledApps/IBM_Payments.ear/Payments.war** directory of your WebSphere Commerce Payments installation.

For **iSeries**

Note: Javadoc is provided for the Cashier in the /QIBM/ProdData/HTTP/Protect/PymSvr/Doc/API directory of your WebSphere Commerce Paymentsinstallation. It can also be accessed through `http://<hostname>:2001/QIBM/PymSvr/Doc/API/index.html`. The *ADMIN HTTP server instance must be started using the Start TCP/IP Server (STRTCPSVR) CL command before accessing this URL.

Building profiles

In “Designing your integration” on page 20, you chose which profiles to make available on your site. This may have included writing your own profiles. The next step is to edit these profiles for use with your merchant software. This part can be broken into several parts.

WebSphere Commerce Paymentsconfiguration

If your integration will use multiple WebSphere Commerce Paymentsinstances, then you might choose to store your WebSphere Commerce Payments configuration information within your profiles. To do this, you must supply the PaymentsConfiguration element in your profiles. This element indicates the location of your WebSphere Commerce Payments instance, the userid and password to use for this instance, whether or not to use SSL, and (optionally) socks server information.

Parameters and SelectStatements

When you have a complete list of WebSphere Commerce Payments and cassette parameters to provide in your profile, you must specify where each parameter will get its value. The value can come from one of four sources: a hard-coded constant in the profile, a value from the order processing environment passed on the `collectPayment()` or `issueCommand()` call, a field in a relational database, or it may be calculated by Cashier extension code. For each parameter in the profile, you must define where the relevant value can be found in your merchant software. (Your

merchant software may publish a formal definition of its interface, which provides a list of WebSphere Commerce Payments parameters and the locations of their values in that merchant software.)

For example, if the Cashier will run on a system in which there is only one merchant, then it would make sense to hard-code the MERCHANTNUMBER parameter in your Cashier profiles:

```
<Parameter name="MERCHANTNUMBER"><CharacterText>1</CharacterText></Parameter>
```

To specify that the value for the WebSphere Commerce Payments ORDERNUMBER parameter is included in the Map passed on `issueCommand()` (associated with the key `orderNum`), include the following:

```
<Parameter name="ORDERNUMBER"><CharacterText>{orderNum}</CharacterText></Parameter>
```

To specify that the values for the WebSphere Commerce Payments AMOUNT and CURRENCY parameters will be retrieved from a relational database, include the following:

```
<SelectStatement id="sql1">SELECT AMT, CUR FROM ORDER_TABLE WHERE ORDERNUMBER=
{orderNum}</SelectStatement>
<Parameter name="AMOUNT"><DatabaseValue statementID="sql1" columnName="AMT">
</Parameter>
<Parameter name="CURRENCY"><DatabaseValue statementID="sql1" columnName="CUR">
</Parameter>
```

For this example, the amount and currency for the order are retrieved from the ORDER_TABLE in the columns called AMT and CUR, respectively. Note that the parameters reference a SelectStatement which provides a row of data for a single order. **orderNum** in the SelectStatement must be provided in the data passed to `collectPayment()` or `issueCommand()`.

To specify that the value for the WebSphere Commerce Payments ORDERURL parameter will be constructed in a Cashier extension class named URLBuilder, include the following:

```
<Parameter name="ORDERURL"><ExtensionValue name="URLBuilder"></Parameter>
```

URLBuilder must be a Java class which implements the CashierExtension interface. URLBuilder.class must be placed in your classpath.

Buy Page information

In “Designing your integration” on page 20 there are descriptions of some ways in which your integration may use the BuyPageInformation element of your profiles. Based on your integration design, you must provide information which will make the generation of buy pages possible. If your merchant software supports shopping in multiple languages, then you should give extra consideration to localization issues on the buy page.

When you have finished editing your profiles, it is recommended that you save them with file names that conform to the following convention: .

```
MerchantSoftwareNameCassetteName.profile
```

For example, the WebSphere Commerce Payments provides a profile for use with the SampleCheckout servlet and the OfflineCard cassette, which is called `SampleCheckoutOfflineCard.profile`.

Including necessary files

To use Cashier, include the following files in the classpath:

- etillCal.zip
- xml4j.jar
- eTillxml4j209.jar
- ibmjsse.jar. This file is only needed if using ssl support for communication with the WebSphere Commerce Payments.
- a JDBC driver. This is only needed if using Database Value parameters.
- any extension classes referenced in the Cashier's profiles

You should provide all the Cashier's profiles and the profile.dtd in a single directory.

Creating a Cashier object

There are three ways to construct a Cashier object:

1. You can construct a Cashier without specifying any information about the Cashier-to-WebSphere Commerce Payments communication channel (such as the WebSphere Commerce Payments hostname and port, and whether or not to use SSL). With this method, the cashier profiles must include the <PaymentsConfiguration> element which includes the needed information.

   ```
   public cashier(String profileDirectory) throws Cashier Exception
   ```

2. Alternately, you can specify WebSphere Commerce Payments configuration information on the constructor of the Cashier. In this case, it is not necessary to include <PaymentsConfiguration> element in your profiles. However, if you do include <PaymentsConfiguration> in a profile, it will override the constructor-provided configuration information.

   ```
   public cashier(String profileDirectory,
    String paymentsHostname
    String paymentsPort
    String userid
    String password
    boolean useSSL) throws CashierException
   ```

3. You can use the following constructor which allows you to connect to WebSphere Commerce Payments via a socks server.

   ```
   public cashier(String profileDirectory,
    String paymentsHostname,
             int paymentsPort,
             String socksHostname,
                  int socksPort,
                  String userid,
                  String password,
                  boolean useSSL) throws CashierException
   ```

You should decide which method to use based on the design of your integration. For example, if you use a single WebSphere Commerce Payments and you frequently change the administrator's password, it would be easier to provide WebSphere Commerce Payments configuration on the Cashier's constructor rather than having to update PaymentsConfiguration elements in each of your profiles.

The Cashier can be safely used in a multi-threaded environment. Internally, it maintains a cache of profiles and consequently you can optimize your integration by reusing the same Cashier instance (rather than repeatedly instantiating new Cashier objects).

CollectPayment

The CollectPayment element contains all the data needed to create WebSphere Commerce Payments orders using the cashier.

```
<Profile useWallet="false">
  <CollectPayment>

    ...

  </CollectPayment>
</Profile>
```

Creating orders in the WebSphere Commerce Payments – issueCommand()

To create orders in the WebSphere Commerce Payments, call the cashier's `issueCommand()` method. The arguments of the `issueCommand()` method include:

- *command* is the constant indicating which command you wish to issue. See the Cashier class documentation for the list of allowed commands. To create orders, the command must be ACCEPTPAYMENT or RECEIVEPAYMENT.
- *profileName* is the name of the profile which used to create the WebSphere Commerce Payments command.
- *locale* is the locale in which your merchant software is presenting text to a shopper (optional)
- values from the order processing environment
- a database connection (optional)
- *queryable* is an optional querying interface, which can be used to return a list of values for a parameter using a Hashtable.

The determination of which profile should be used is typically based on the buyer's choice of payment method. For example, you may provide a profile for Cash On Delivery orders and another for credit card orders. The locale which you supply to `issueCommand()` should match the locale in which the buyer is shopping. The WebSphere Commerce Payments will use this value to create a localized message which may be displayed to the shopper in the event of an error. This allows for new cassettes to be added without the merchant software having to construct its own error messages.

Any information that is available when your merchant software is processing orders should be passed to the Cashier. Depending on your merchant software, this may include the parameters which constitute an HTTP request, name-value pairs, or others. Any data used by your profiles should be put in a Map and passed on the `issueCommand()` call.

An initialized JDBC Connection should be supplied if your profiles include any DatabaseValue parameters. Note that the Cashier will not close the JDBC Connection during the `issueCommand()` call.

Checking the status of an order – checkPayment()

The Cashier provides a simple method called `checkPayment()` to query the status of an order that you have created. A call to `checkPayment()` will return a CheckPaymentResponse object which contains the state of the order. The Cashier javadoc (provided with the WebSphere Commerce Payments) describes the possible values which this state can have.

```
CheckPaymentResponse checkPaymentResponse =
cashier.checkPayment(merchantNumber, orderNumber);


if (checkPaymentResponse.getPrimaryReturnCode()   == 0 &&
    checkPaymentResponse.getSecondaryReturnCode() == 0)
{
  switch (checkPaymentResponse.getState())
  {
    case CheckPaymentResponse.APPROVED:
      ...
    case CheckPaymentResponse.MISSING:
      ...
    ...
  }
}
else
{
  ...
}
```

Using BuyPageInformation

Buy Page Information, as defined by your integration design, can be retrieved by calling the Cashier's getBuyPageInformation() method. To retrieve the Buy Page Information reference, you call getBuyPageInformationReference().

Tracing

The Cashier provides a trace mechanism to aid in the writing of your integration. A trace class (SimpleCashierTrace) is provided in etillCal.zip which will write to a file. Alternatively, by implementing the CashierTrace interface, it is possible to have the Cashier use your merchant software's existing trace classes.

Tracing can be enabled on a per-profile basis to help diagnose problematic profiles. Tracing for a profile is enabled by setting enableTrace=″true″ in the Profile element.

```
Cashier cashier = new Cashier("d:\\cashierProfileDirectory");
SimpleCashierTrace simpleCashierTrace =
new SimpleCashierTrace("d:\\cashierLogDirectory");
cashier.setTraceClass(simpleCashierTrace);
```

Exceptions

When an error occurs in the Cashier, an exception is thrown indicates the source of the problem. There are two varieties of exceptions in the Cashier: ProfileExceptions and CashierExceptions. The Cashier throws a ProfileException when the Cashier encounters a profile that is not well-formed, is not valid, or has logical errors that prevent it from being used to create orders. CashierExceptions are thrown when the Cashier is used improperly or when there is an error accessing the merchant database.

When calling the Cashier, you should be aware that `issueCommand()`, `collectPayment()` and `checkPayment()` throw PaymentServerCommunicationExceptions. This provides you the opportunity to write retry logic around these calls.

CashierExceptions and ProfileExceptions may contain a Throwable object which will provide further details of the error. Both of these exception provide a method called getNestedException() to provide access to this Throwable object.

Writing extensions

Values for most WebSphere Commerce Payments parameters can be obtained using constants, values from your order processing environment, or values from your databases. However, there may be some parameters for which the value can not be so easily derived. For example, if a parameter requires a textual description of the shopper's order and your merchant software doesn't contain that description in the proper format, then you may need to code a cashier extension to build the proper value for this parameter.

A Cashier Extension is code that is run by the cashier to build a value for a WebSphere Commerce Payments parameter. To write a Cashier Extension, you must write a class which implements the CashierExtension interface. This interface contains only a single method – getValue(). getValue() is called in a Cashier Extension when using a cashier profile which contains an ExtensionValue parameter references that extension.

```
public class SampleExtension implements CashierExtension
{
  public String getValue(String keyword, Hashtable environmentValues,
Hashtable PaymentsParameters, Connection connection,
CashierTrace cashierTrace, Locale locale) throws CashierException
  {
    if (keyword.equals("$DATETIME"))
    {
      Date date = new Date();
      return date.toString();
    }
    else if (keyword.equals("$RANDOMNUMBER"))
    {
     return String.valueOf(Math.random());
    }
    else ...
  }

}
```

SampleCheckout application

SampleCheckout is a sample application that demonstrates how applications can use the Cashier to integrate with WebSphere Commerce Payments. The application uses an HTML interface that can be accessed from the URL http://hostname/webapp/SampleCheckout. Source code is provided in the samples/SampleCheckout directory.

For **iSeries**, source code is provided in the /QIBM/ProdData/PymSvr/Samples/SampleCheckout directory.

WebSphere Commerce Payments

Merchant number	123456789
Order number	
Amount	
Currency	

Payment method	(•) Offline Card ( ) Cash on delivery (via CustomOffline) ( ) Bill me (via CustomOffline)

Brand	
Credit card number	
Expiration date	

Buy

Overview

SampleCheckout is a simple order entry system that allows orders to be created using different payment methods. Users must enter basic order information - such as order number, merchant number and order amount - as well as the payment information used to collect payment for the order. SampleCheckout allows the configuration of any number of different payment methods; each payment method is supported by a Cashier profile. The SampleCheckout profiles contain the HTML needed to build the payment information part of the buy page as well as the data needed to build an API request to create the order in WebSphere Commerce Payments.

SampleCheckout attempts to display the buy page using the language preference of the user's browser. SampleCheckout is translated into the same languages supported by WebSphere Commerce Payments. If a user's language preference is not supported, the buy page is presented in English. To select a language for Internet Explorer, click **Tools** from the menu bar, then **Internet Options**, and then click the **Languages** button. From Netscape Navigator, click **Edit** from the menu bar, then **Preferences**, then select **Language** under the Navigator category. Fields marked with a red asterisk are required input. Others are optional.

SampleCheckout works for both Cashier profiles that do not use a wallet and those that do. If profiles do require a wallet, SampleCheckout assumes that the ReceivePayment API response from the WebSphere Commerce Payments contains an HTTP wallet wake up message.

Requirements

The SampleCheckout application uses Java Server Pages (JSP) and dynamic HTML technologies. It is automatically installed and configured for the Offline Card and the CustomOffline cassettes when WebSphere Commerce Payments is installed. To use it, you must have a servlet engine that supports JSP (such as WebSphere Application Server) and a browser that supports dynamic HTML. It has been tested with Netscape Navigator version 4 and Microsoft Internet Explorer versions 4 and 5.

The Web Server must be configured to serve the files in the directory web/SampleCheckout in response to URIs beginning /webapp/Payments/SampleCheckout.

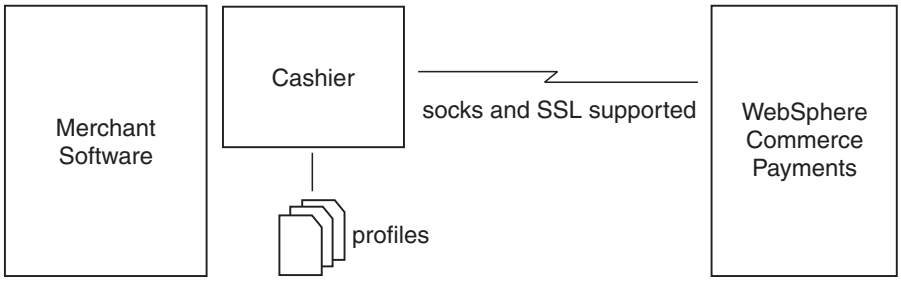
A servlet called SampleCheckoutServlet must be defined to the servlet engine with a classpath containing the following: eTillCal.zip, xml4j.jar and ibmjsse.jar.

For **iSeries**, the classpath must contain eTillCal.zip and xml4j.jar.

Also required is the samples/SampleCheckout directory (which includes all the necessary Java class files, properties files and the SampleCheckout.xml along with its SampleCheckout.dtd file).

The SampleCheckout.xml configuration file must point to valid profile and log directories and the configured payment options must point to valid Cashier profiles.

Configuration



SampleCheckout uses a configuration file named samples/SampleCheckout/SampleCheckout.xml to define the following:

- the WebSphere Commerce Payments configuration information (hostname, port, use of socks and SSL) Since SampleCheckout provides a global method for storing WebSphere Commerce Payments configuration information, it is not necessary for each profile to specify a PaymentsConfiguration element.
- the directory containing the SampleCheckout profiles — By default, this is the profiles directory where WebSphere Commerce Payments is installed.
- the directory where Cashier trace information is written — By default, this is the log directory where WebSphere Commerce Payments is installed.
- the available payment methods and which Cashier profiles each method uses — New payment methods can be supported by SampleCheckout by adding the following element to the PaymentOptionList element of SampleCheckout.xml:

```
<PaymentOption id="newmethod" profile="newprofname">
   New Payment Method
</PaymentOption>
```

where newmethod is the ID of this new payment method, newprofname is the name of the Cashier profile that supports the method, and ″New Payment Method″ is the label that is displayed on the SampleCheckout buy page.

- the currencies supported by SampleCheckout

SampleCheckout Profiles

When WebSphere Commerce Payments is installed, SampleCheckout profiles is installed in the profiles directory. SampleCheckout profiles must contain a BuyPageInformation element and the parameter definitions for all the parameters required by WebSphere Commerce Payments and the specified cassette for the profile. SampleCheckout profiles do not need to contain WebSphere Commerce Payments configuration information since this is specified in the SampleCheckout configuration file. However, if a PaymentsConfiguration element is specified, then it will override the configuration specified in SampleCheckout.xml.

Buy page information

The BuyPageInformation element in each profile must contain the HTML to create the payment information section of the buy page. Each profile's BuyPageInformation contents is inserted into the HTML <table> and <form> tags as follows:

```
<FORM NAME="cassetteform" METHOD="POST"
ACTION="/webapp/Payments/SampleCheckout">
  <TABLE BORDER="0" WIDTH="100%" CELLSPACING="1" CELLPADDING="2">

     ... <BuyPageInformation> contents ...

  </TABLE>
</FORM>
```

SampleCheckout provides localization support for the contents of the BuyPageInformation element via Java ResourceBundle files. These files contain a mapping of keywords to text and this allows writers of SampleCheckout profiles to avoid hard coding text in the BuyPageInformation elements. Instead, at run-time, SampleCheckout replaces the keywords enclosed in curly braces with text from the ResourceBundle for the user requested language. The name of the ResourceBundle used by SampleCheckout BuyPageInformation elements is indicated by the reference attribute. For example, if a SampleCheckout profile contains the following elements:

```
<BuyPageInformation reference="SampleCheckoutOfflineCard"
  ...
  <p>{BPMESSAGE}</p>
  ...
<BupPageInformation>
```

and the user has requested buy pages in Canadian French, then the application will search for localized text for BPMESSAGE in the following Java ResourceBundles until it finds a match.

SampleCheckoutOfflineCard_fr_CA.class
SampleCheckoutOfflineCard_fr_CA.properties
SampleCheckoutOfflineCard_fr.class
SampleCheckoutOfflineCard_fr.properties
SampleCheckoutOfflineCard.class
SampleCheckoutOfflineCard.properties

Profile environment variables

The following table defines the variables that SampleCheckout makes available to its profiles. These variables can be used in a profile by enclosing the variable name

in curly braces. For example, {merchantnumber} is replaced by the merchant number entered by the user on the buy page.

Variable Name	Content
merchantnumber	the merchant number entered on the form
ordernumber	the order number entered on the form
currency	the 3-digit number for the ISO 4217 currency selected on the form
currencyAlpha	the 3–letter alphabetic value for the ISO 4217 currency
amount	the amount entered on the form
amountLowestCurrUnits	the amount value, converted to the currency's lowest units; for example, 5 US dollars converts to 500 cents.
paymentOption	the Payment method selected on the form
other form variables as specified in the BuyPageInformation element of each profile	the value entered on the form

Chapter 4. Client API Library (CAL)

Merchant business software can issue payment, administration, and query commands to the WebSphere Commerce Payments. Requests are sent to the WebSphere Commerce Payments by issuing HTTP POST messages, and responses are received from the WebSphere Commerce Payments in the form of XML documents embedded in the HTTP. The Java Client API Library (CAL) provides a Java programming interface that enables merchant software written in Java to issue these commands to the WebSphere Commerce Payments and receive the responses. CAL provides a wrapper that shields the merchant software writer from having to understand the details of HTTP communications and XML document parsing. CAL provides Java objects that allow a merchant program to:

- create requests to be sent to the WebSphere Commerce Payments
- communicate with the WebSphere Commerce Paymentsvia a TCP connection or a SSL connection
- process responses from the WebSphere Commerce Payments

A merchant program written to use CAL has several steps:

- Create a PaymentServerClient
- Issue commands to the WebSphere Commerce Payments
  - create a Hashtable object and populate it with keyword/value pairs
  - issue the command
  - process the return codes
  - process the returned data
- Close the PaymentServerClient

The remainder of this section describes these steps at a high level. Details can be found in the JavaDoc, which is located in this location:

- For Windows® and UNIX platforms: **<Payments_installdir>/WebSphere/AppServer/InstalledApps/IBM_Payments.ear/Payments.war** directory.
- For iSeries: **/QIBM/ProdData/HTTP/Protect/PymSvr/Doc/API** directory. It can also be accessed through **http://[hostname]:2001/QIBM/PymSvr/Doc/API/index.html**. The *ADMIN HTTP server instance must be started using the Start TCP/IP Server (STRTCPSVR) CL command before accesssing this URL.

CAL Public Classes

CAL is structured as a Java Class library with a number of public classes. These classes can be divided into three groups:

1. **Client classes**: A merchant program will create one instance of these classes to communicate with the WebSphere Commerce Payments.
   - `PaymentServerClient`: Communicate with the WebSphere Commerce Payments over a TCP connection (with or without SOCKS support)
   - `PaymentServerSSLClient`: Communicate with the WebSphere Commerce Payments over an SSL connection
2. **The Response class**:Each command issued to the WebSphere Commerce Payments returns an instance of this class: PaymentServerResponse

3. **The PSObject classes**: Data returned from Query commands is processed into a set of PSObjects reflecting the actual WebSphere Commerce Payments objects.
   - `PSObject`: superclass of all these objects
   - `PSAdminObject`: superclass of all administration objects
   - `PSOrder`: represents an Order
   - `PSPayment`: represents a Payment
   - `PSCredit`: represents a Credit
   - `PSBatch`: represents a Batch
   - `PSBatchTotal`: represents batch totals for a particular currency
   - `PSPaymentServer`: represents the Payment Server administration object
   - `PSMerchant`: represents a Merchant administration object
   - `PSCassette`: represents a Cassette administration object
   - `PSMerchantCassetteSettings`: represents a `PaymentSystem` administration object
   - `PSAccount`: represents an Account administration object
   - `PSCassetteObject`: represents an object attached by a cassette to a generic object
   - `PSCassetteConfigObject`: represents an administration object attached by a cassette to a generic administration object
   - `PSAbout`: provides version information for the WebSphere Commerce Paymentsand the user name of the person issuing the command
   - `PSCassetteAbout`: provides version information for a WebSphere Commerce Paymentscassette

Creating a PaymentServerClient

A `PaymentServerClient` represents a connection from the merchant program to the WebSphere Commerce Payments. It is a persistent object, designed to be created at the beginning of a merchant program, used and reused throughout that program and closed when the program terminates. The `PaymentServerClient` has a single socket connection that it maintains until the `PaymentServerClient` is closed. The `PaymentServerClient` can be created in several ways to reflect different communication options.

A basic `PaymentServerClient` is constructed with three arguments:

```
PaymentServerClient (String dtdPath, String hostName, int portNumber)
```

This constructor creates a client that will communicate using TCP to WebSphere Commerce Payments at hostName:portNumber. Two additional arguments, socksHost and socksPort, can be added to the basic constructor. This will create a client that communicates to the WebSphere Commerce Payments through a SOCKS server.

```
PaymentServerclient(String dtdPath, String hostName, int portNumber,
String socksHost, int socksPort)
```

Two additional constructors allow the specification of a hashtable to be used to specify additional keyword/value pairs to be passed in the HTTP header.

```
PaymentServerClient (String dtdPath, String hostName, int portNumber,
Hashtable httpHeaderFields)
PaymentServerClient (String dtdPath, String hostName, int portNumber,
String socksHost, int socksPort, Hashtable httpHeaderFields)
```

Other communication options are created with subclasses of PaymentServerClient. A PaymentServerSSLClient communicates with the WebSphere Commerce Payments over an SSL connection.

```
PaymentServerSSLClient(String dtdPath, String hostName, int portNumber)
PaymentServerSSLClient(String dtdPath, String hostName, int portNumber,
  String socksHost, int socksPort)
PaymentServerSSLClient(String dtdPath, String hostName, int portNumber,
  Hashtable httpHeaderFields)
PaymentServerSSLClient(String dtdPath, String hostName, int portNumber,
  String socksHost, int socksPort, Hashtable httpHeaderFields)
```

Note: The DTDPath specified when the PaymentServerClient is instantiated is used throughout the session (for all subsequent commands processed before the close ()). The DTDPath on the PaymentServerClient is optional and can be NULL but better performance can be realized if the DTDPath is specified.

Preparing the iSeries for SSL Support

Note: These instructions are for **iSeries** only.

To prepare your system to use Secure Sockets Layer (SSL), you must install the Digital Certificate Manager Licensed Program: 5722–SS1 OS/400 — Digital Certificate Manager

You must also install one of the following Cryptograpahic Access Provider Licensed Programs:

- 5722-AC2 Cryptographic Access Provider 56-Bit
- 5722-AC3 Cryptographic Access Provider 128-Bit

If client authentication is required by the server, you may set the following properties to specify which digital certificate to use:

- os400.certificateContainer
- os400.certificateLabel

If these properties are not set, the default system certificate (if any) will be used. More information on iSeries documentation to install Java/SSL is found at: **http://publib.boulder.ibm.com/pubs/html/as400/infocenter.html**. Follow the link for the current iSeries version, then select: **Programming**, **Java**, **iSeries Development Kit for Java**, **Security**, **Secure Sockets Layer**.

Issuing WebSphere Commerce Paymentscommands

The issueCommand method of PaymentServerClient is used to send commands to the WebSphere Commerce Payments. There are several overloaded versions of the issueCommand method. At a minimum, each issueCommand method takes the following parameters:

WebSphere Commerce Payments API command name
The name of the WebSphere Commerce PaymentsAPI command name. See Chapter 7, “WebSphere Commerce Payments command reference” on page 59 for a list of WebSphere Commerce Payments API commands. The public interface PaymentCommandConstants defines constants for each API command. Refer to the JavaDoc for details.

A hashtable of the keyword/value pairs to be sent with the WebSphere Commerce Paymentscommand

This Java Hashtable represents the parameters to be passed with the specified API command. The keys to the hashtable are Strings that represent the API command parameter name. The values represent the value of the API command parameter. The values can be one of these types:

- String: a Unicode string in all supported character sets
- byte[]: a byte array, for binary data
- Integer: a 4-byte integer
- Date: a Java Date (java.util.Date) representing a timestamp
- Boolean: a boolean value
- Vector: a vector of any of the above. Vector values are a special case. If a keyword is assigned a Vector of values, it will be included in the HTTP body multiple times, one for each entry in the Vector.

It should be noted that CAL does not check these keyword/value pairs to ensure they are valid for the specified keyword, or that the data types of the values are correct. CAL simply passes all keyword/value pairs on to the WebSphere Commerce Payments. See Chapter 7, “WebSphere Commerce Payments command reference” on page 59 for a list of required and optional parameters for each WebSphere Commerce Paymentscommand. The public interface `PaymentCommandConstants` defines constants for each API command parameter value. Refer to the JavaDoc for details.

Authentication information

When WebSphere Commerce Payments receives a command, it authenticates the user through the use of realms. When writing programs using WebSphere Commerce Payments, it must be understood which realm the WebSphere Commerce Payments is using. The realm contains the list of users that are potentially authorized to access WebSphere Commerce Payments, along with their authentication information. The realm dictates whether or not each command should have a userId/password associated with it, or, more generally, a byte array that contains other authentication credentials. Therefore, look at your realm documentation to determine which of these `issueCommand` methods you should use. The default realms that are provided with WebSphere Commerce Payments all use userId/password for authentication. If you don’t know what realm your WebSphere Commerce Payments is using, you can look at your WebSphere Commerce Payments Settings screen in the user interface.

The basic versions of the `issueCommand` method are:

```
issueCommand(String command, Hashtable keywordValuePairs, String basicAuthUserid,
  String basicAuthPassword)
issueCommand(String command, Hashtable keywordValuePairs, byte pmAuthObject[])
```

In addition, there are versions of the `issueCommand` method that allow the specification of a hashtable to be used to specify additional keyword/value pairs to be passed in the HTTP header:

```
issueCommand(String command, Hashtable keywordValuePairs, Hashtable httpHeaderPairs,
String basicAuthUserid, String basicAuthPassword)
issueCommand(String command, Hashtable keywordValuePairs, Hashtable httpHeaderPairs,
byte pmAuthObject[])
```

`issueCommand` will throw an exception in the event of errors or other processing problems.

Specifying additional information in the HTTP Header

There are two ways to specify additional information in the HTTP Header:

- in the constructor of the PaymentServerClient object, which causes the additional parameters to be specified on all commands issued to the WebSphere Commerce Payments
- in the `issueCommand` method, which causes the additional parameters to be specified only for the command that is being issued, thus allowing the HTTP Header to be tailored for each WebSphere Commerce Payments command. An example of this occurs on the **AcceptPayment** and **ReceivePayment** API commands. For these commands, the WebSphere Commerce Payments will return message text when the processing of the command was not successful. The message text provides additional information in the language preference of the client application as specified by the `PM-Accept-Language` tag in the HTTP header. If the `PM-Accept-Language` tag is not specified in the HTTP Header, then the default language of the machine running the servlet is used. See the `PaymentServerResponse` methods `getBuyerMessage()` and `getMerchantMessage()` for additional information regarding these messages. In addition, CAL provides a convenience method to create the keyword/value pair for the `PM-Accept-Language` tag. See the `PaymentServerClient` method `addLocaleToHttpHeader` for details.

Processing responses from WebSphere Commerce Payments

A `PaymentServerResponse` object is returned by `issueCommand`. This object contains methods that allow the merchant software to access the primary and secondary return codes that were returned as a result of issuing the command to the WebSphere Commerce Payments. See Appendix A, “WebSphere Commerce Payments return codes” on page 129 for a list of WebSphere Commerce Payments return codes. If a programmatic error occurs, a Java exception is thrown. There are two types of exceptions defined in CAL:

- `PaymentServerCommunicationException` This exception is thrown when CAL is having trouble communicating with the WebSphere Commerce Payments. Possible causes include:
  - CAL received a bad HTTP response; this generally means that something is wrong with the Payment Servlet or the WebServer/WebSphere configuration.
  - CAL took an IOException, which means that the TCP layer or the SSL layer threw an IOException (for example, could not connect to the WebSphere Commerce Payments, or the connection went down prematurely). If this exception results from an IOException, the IOException is stored within the `PaymentServerCommunicationException` (and can be accessed by the merchant programmer).
- `PaymentServerClientException` This is an internal exception thrown by CAL. It indicates a defect in CAL.

Process returned objects

When a command results in returned data (for example, Query commands), a set of `PSObjects` is returned as part of the `PaymentServerResponse`. These objects correspond to basic WebSphere Commerce Payments objects. The interpretation of these fields can be found in Chapter 8, “WebSphere Commerce Payments data” on page 111.

The `PaymentServerResponse` object contains the method `getObjectCount` which returns the number of objects that were returned in the response. This is especially useful for queries using RETURNATMOST, which limits the size of data.

Closing the PaymentServerClient

The `PaymentServerClient` classcontains a `close()` method. Merchant programs should call `close()` prior to exiting. This is not particularly important for simple programs using standard TCP or SOCKS communication because the Java Runtime will clean up these resources on exit. However, it is extremely important for SSL clients. Failure to call `close()` on these clients can result in problems when the merchant's application is restarted. Since merchant programs can be converted to use SSL at any time, it is good practice to ensure that `close()` is called in all cases.

Sample CAL program

This section contains a skeleton of a simple CAL program. Sample CAL programs are available and are located in the following locations:

- For Windows and UNIX platforms:
  - **<Payments_installdir>/samples** directory.
- For iSeries:
  - **/QIBM/ProdData/PymSvr/CAL/Samples**directory.

A merchant program written to use CAL has three primary steps:

1. Create a PaymentServerClient
2. Issue commands to the WebSphere Commerce Payments
   a. Create a hashtable and populate it with keyword-value pairs
   b. Issue the command
   c. Process the return codes
   d. Process the returned data
3. Close the PaymentServerClient

An example CAL program follows:

```
PaymentServerClient client = new PaymentServerClient(dtdPath, hostName,
port);

while (...)
  {
    Hashtable keywordValuePairs = new Hashtable();
    keywordValuePairs.put("merchantnumber","123456789");
    ... using documentation in the programmer's reference as a guide, fill
 in other keywordValuePairs ...

    PaymentServerResponse response =
client.issueCommand(command,keywordValuePairs,userid,password);

    int primaryRC = response.getPrimaryRC();
    int secondaryRC = response.getSecondaryRC();
    ... process return codes ...
    String contentType = response.getContentType();
    if (contentType != null)
    ... process contentType

    Enumeration objects = response.getObjects();
    while (objects.hasMoreElements())
      {
        PSObject object = (PSObject) objects.nextElement();
        ... process object ...
      }
   }

client.close();
```

Installing Files Required by CAL

All files required by CAL can be found in the following locations:

- For Windows and Unix platforms:
  - Files can be found in a zipped file called eTillCal.zip, which is found in the `\WebSphere\AppServer\installedApps\IBM_Payments.ear` directory. To install the required files, unzip eTillClientSDK.zip to a directory of your choice. Recommended directory names are C:\PaymentsClient for Windows NT® or /usr/PaymentsClient for AIX®, Solaris and Linux®.
- For iSeries:
  - Files required by CAL are installed at /QIBM/ProdData/PymSvr/CAL/etillCal.zip and /QIBM/ProdData/PymSvr/XML/eTillxml4j209.jar. No additional installation steps are required.

Be sure to include the required class libraries in the CLASSPATH environment variable for the system or for the session in which your WebSphere Commerce Payments application will run.

For Machines that don't have WebSphere Commerce Payments Installed

If you plan to write to the CAL interface or execute CAL programs from a machine that does not have the WebSphere Commerce Payments installed, perform the following steps:

For Windows and UNIX platforms:

1. From a machine where the WebSphere Commerce Payments is installed, copy the following files to your machine. These files can be found in the WebSphere Commerce Payments directory:
   - *etillCal.zip*
   - *eTillxml4j209.jar*
   - *ibmjsse.jar* (Only required at runtime if you are using SSL.).
2. If you will be using the IBM-supplied DTD, copy the *IBMPaymentServer.dtd* file from the WebSphere Commerce Payments */include* directory.
3. Edit your system CLASSPATH to include *etillCal.zip* and *eTillxml4j209.jar*.

For iSeries:

1. From a machine where the WebSphere Commerce Payments is installed, copy the following files to your machine. These files can be found in the WebSphere Commerce Payments root directory:
   - */QIBM/ProdData/PymSvr/CAL/etillCal.zip*
   - */QIBM/ProdData/PymSvr/XML/eTillxml4j209.jar*
2. If you want to use SSL through CAL running on another iSeries system, you will also need the Licensed Programs listed in the ″Preparing iSeries for SSL Support″ section on the remote iSeries system.
3. If you will be using the IBM-supplied DTD, copy the *IBMPaymentServer.dtd* file from the */QIBM/ProdData/PymSvr/XML/DTD* directory.
4. Edit your system CLASSPATH to include *etillCal.zip* and *eTillxml4j209.jar*.

Note: If you want SSL support through CAL running on a non-iSeries system, copy ibmjsse.jar.

Chapter 5. Event notification

The WebSphere Commerce Payments provides an event notification service to enable merchant software (or non-merchant software such as network management systems) to listen for events and perform appropriate actions in the merchant's business system (For instance, delivering an order to the shipping department when an event indicates that an order has been approved). One function of this service is as a performance optimization for systems that normally issue Query commands to determine the state of WebSphere Commerce Payments objects. By listening for the events that occur when object states change, a merchant system can react quickly without incurring the full overhead of a polling loop. In addition, the event notification service can be used by network management software to monitor the health of the WebSphere Commerce Payments.

Merchant software registers its interest in WebSphere Commerce Payments events and specifies a URL. When events occur, the event notification service sends an HTTP POST to a destination specified by the URL. The merchant software should be responsible to receive the events. The merchant software that listens for these events can be a CGI, Java Servlet or a program which listens to the port specified in the registration.

Event types and contents

The WebSphere Commerce Payments event notification service defines and will send the following three types of events:

1. **State change event**. These events are sent when the state of a Framework object has been changed. For example, the state of an Order object is changed from ″Received″ to ″Approved″.
2. **Cassette-specific event**. The cassette can use this event type to notify merchants of events that occur within the cassette. The cassette defines the content of the event. Not all cassettes will implement cassette-specific events.
3. **Network management event**. These events are sent when the WebSphere Commerce Payments is started or stopped.

WebSphere Commerce Payments provides the ″state change event″ for the Framework financial objects and the Framework up and down network management events. The merchant software should refer to the appropriate Cassette Supplement to find out which cassette events are being supported.

Every event contains the following ″basic″ contents:

- **EventType**: The type of event.
- **Timestamp**: Time when the event happens.
- **ObjectID**: Identifies the object which the event is referring to. The ObjectID may consist of several fields.

Different event types may contain different information, which is described in the next section.

State change event

State change event

Name	Value

State change event

EventType	″1″
Object	One of the following values: • Order • Payment • Credit • Batch
<ObjectID>	The ObjectID is dependent on the Object type. Each object is identified by a set of keys. (For example, an Order is identified by its MerchantNumber and OrderNumber.)
PreviousState	State name. See “WebSphere Commerce Payments payment objects” on page 111 for state definitions.
Current State	State name. See “WebSphere Commerce Payments payment objects” on page 111 for state definitions.
TransactionId	Transaction identifier that was supplied by the user on the AcceptPayment or ReceivePayment API.
OrderData1	Auxiliary data that was supplied by the user on the AcceptPayment or ReceivePayment API.
OrderData2	Auxiliary data that was supplied by the user on the AcceptPayment or ReceivePayment API.
OrderData3	Auxiliary data that was supplied by the user on the AcceptPayment or ReceivePayment API.
OrderData4	Auxiliary data that was supplied by the user on the AcceptPayment or ReceivePayment API.
OrderData5	Auxiliary data that was supplied by the user on the AcceptPayment or ReceivePayment API.

Cassette-specific event

For cassette-specific events, in addition to the name-value pairs defined in the following table, each cassette can define its own name-value pairs. The documentation for each cassette will detail the cassette-specific name-value pairs and the rules which define when these events are sent.

Cassette-specific event

Name	**Value**
EventType	″2″
CassetteName	<CassetteName> value in ASCII character string.
MerchantNumber	Integer in ASCII characters.

Network management event

Network management events

Name	**Value**
EventType	″3″

Network management events

ComponentName	Either one of the following values in ASCII character string: • Framework • <CassetteName>
Status	Either one of the following integer values in ASCII string: • ″1″: (Denotes running) • ″2″: (Denotes not running)

For example, the WebSphere Commerce Payments will send a State Change Event with the following contents to the Event Listeners:

```
EVENTTYPE=1
TIMEGENERATED=
MERCHANTNUMBER=
PREVIOUSSTATE=
CURRENTSTATE=
OBJECT=
ORDERNUMBER=
PAYMENTNUMBER=
CREDITNUMBER=
BATCHNUMBER=
ACCOUNTNUMBER=
```

WebSphere Commerce Payments will send a Network Management Event with the following contents to the Event Listeners:

```
EVENTTYPE=3
TIMEGENERATED=
COMPONENTNAME=
STATUS=
```

Registering events

In order to receive events, the merchant software must register itself with the WebSphere Commerce Payments. There are two types of event listeners: merchant and non-merchant. Merchant listeners can only register merchant-specific events (all state-change and cassette-specific events). Non-merchant software, such as a network management system, can only register a network management event. The merchant and the non-merchant software can register the same type of events multiple times. In this case, the events will be broadcast to each of the registered locations.

The API commands for registering and managing event listeners are discussed in Chapter 7, “WebSphere Commerce Payments command reference” on page 59.

Event ListenerURL parameter

When creating an event listener, a valid ListenerURL is a required keyword. In the WebSphere Commerce Payments, a valid ListenerURL is defined as a valid Java URL. The same valid Listener URL may have a different format. For example: *http://foo* and *http://foo/* are the same URLs, but *http://foo/xx* and *http://foo/xx/* are two different URLs. The WebSphere Commerce Payments command will convert a valid URL into the WebSphere Commerce Payments canonical URL format, which is a valid URL with the following extensions:

- The WebSphere Commerce Payments command will insert the port number ″80″ if the port number is not defined.
- The WebSphere Commerce Payments command will insert the hostname ″localhost″ if the hostname is not defined.
- The WebSphere Commerce Payments command will insert the hostname ″localhost″ and the port number ″80″ if neither is defined.

By using this canonical URL format, the QueryEventListener command will return the same listener for slightly different input URL strings. For example, if the port number of the listener is 80, then no matter which port number is specified in the URL, the same listener will be returned.

Chapter 6. WebSphere Commerce Payments realm support

WebSphere Commerce Payments authenticates users through the use of realms. A realm is a registry of users along with a single method of authenticating those users (for example, a user's name and password). Examples of realm types include LDAP realms and operating system realms. A user must be defined in a realm before being granted access to resources. Therefore, a user is a valid WebSphere Commerce Payments user if, *and only if*, he is both:

- in the realm
- assigned a role in the WebSphere Commerce Payments

WebSphere Commerce Payments employs a role-based access control scheme which defines four WebSphere Commerce Payments roles:

1. Payments administrator
2. Merchant administrator
3. supervisor
4. clerk

The Payments administrator can use the WebSphere Commerce Payments user interface User's window to assign access (based on role) to a user defined in a realm. Though other realms can be created, the following are provided with the WebSphere Commerce Payments:

- **PSDefaultRealm:** This is a simple realm implementation provided by the WebSphere Commerce Payments which uses a flat file to hold a list of realm users and their corresponding passwords.
- **WCSRealm:** The WCSRealm class is installed automatically by the WebSphere Commerce Payments installation program if WebSphere Commerce Payments is installed on the same machine as WebSphere Commerce Suite. This realm allows the WebSphere Commerce Payments Servlet to use the administrator information that is already registered in the WebSphere Commerce Suite user tables. This administrator information is used for Payments administrators, so that you do not have to define another set of administrator IDs in order to use the WebSphere Commerce Payments user interface. Use this realm when using WebSphere Commerce Payments with WebSphere Commerce Suite
- **PSOS400Realm:** This is a realm implementation provided by the WebSphere Commerce Payments which accesses the iSeries user profiles to validate users and their corresponding passwords. It is the default realm for use with WebSphere Commerce Payments for iSeries.

As previously noted, these are not the only mechanisms for authenticating and authorizing users. Other applications that use the WebSphere Commerce Payments may override WebSphere Commerce Payments realm implementations and use their own support. This chapter provides information on writing new realms for use with the WebSphere Commerce Payments. Note that if your merchant software already shares a user registry with the WebSphere Commerce Payments, writing and implementing a new WebSphere Commerce Payments realm may break the integration between your merchant software and the WebSphere Commerce Payments.

Writing a new WebSphere Commerce Payments realm

WebSphere Commerce Payments is not a stand-alone application and will always work in conjunction with other merchant software. If this software provides a user interface, it is often desirable to provide links into the WebSphere Commerce Payments user interface in order to perform payment operations such as making credits against an existing order, closing batches, or viewing the status of existing payments. Under normal operation, a user would log in to the merchant software and then, when the user clicks a link to get to the WebSphere Commerce Payments user interface, he would need to log in a second time to access WebSphere Commerce Payments. If the merchant software and the WebSphere Commerce Payments both share the same realm, this double login problem can be removed by coding the merchant software links to pass authentication information through the WebSphere Commerce Payments user interface and into the realm code. Single-signon is often an important reason for wanting to write a WebSphere Commerce Payments *plug-in* realm.

This section details the process for writing a new WebSphere Commerce Payments realm. In preparing to write your new realm, you will need to:

- Make decisions about the design of your realm
- Implement a PaymentServletRealm subclass
- Use the trace facility to enable easy problem determination
- Implement single-signon between another user interface and the WebSphere Commerce Payments user interface (that is, provide links between your Web pages and the WebSphere Commerce Payments Web pages using the single-signon function)
- Test the realm

Design points

To write a realm for WebSphere Commerce Payments you must provide a new class that extends the PaymentServletRealm class. You should consider the following design points when writing your new WebSphere Commerce Payments realm:

Realm name

One of the functions your new realm class will provide is getRealmName(). Payments administrators can determine which realm is currently in use by looking at the Realm field of the Basic settings window in the user interface. This realm name is determined by querying the realm using the getRealmName() method. The result is passed to the user interface via the QueryPaymentServer API response. You should provide a getRealmName() method in your realm class that returns a string that can be used by the Payments administrator to understand which realm is in use and, therefore, how he can manage that particular realm.

Realm properties

Your new realm may require specific configuration values. You will determine what these new configuration values are and create new property keywords for them. Your documentation should instruct your users to place these properties in a file called `<realmname>.properties`, where <realmname> is the name of the new realm (e.g. `PSDefaultRealm.properties`) For example, you may need a new property for the hostname of the system where your user registry is held. At start-up time, WebSphere Commerce Payments will read the `<realmname>.properties` file, instantiate your realm, and pass all your realm properties to your realm's init() method.

Authentication mechanism

The WebSphere Commerce Payments API uses HTTP post requests to transfer information. The HTTP request will be made available to your realm's getAuthenticatedUser() method so that you can find the authentication information from the request and determine whether the user issuing the API command is an authenticated member of the realm or not. You must decide how your realm is going to get this authentication information. There are three possible mechanisms for authentication that your realm may use:

1. Basic authentication
2. WebSphere Commerce Payments Authentication Object (that is, PMAUTHOBJECT)
3. Custom authentication

If your realm requires only basic authentication (that is, user ID and password), then WebSphere Commerce Payments supports this via the HTTP authorization header. `PaymentServletRealm` contains a convenience method, getFromHTTPAuthorizationString(), that can be used to read the user ID and password from this HTTP header.

If user ID and password are not sufficient, you can use WebSphere Commerce Payments's PMAUTHOBJECT keyword to pass arbitrary authentication information to the realm. Again, `PaymentServletRealm` provides a convenience method, getPMAuthenticationString(), to allow your realm to retrieve this object from the HTTP request.

Lastly, it is important to note that WebSphere Commerce Payments does not provide any restriction on the mechanism a realm uses to authenticate any HTTP request, so you can decide to implement any authentication method you like.

User interface single-signon

As previously noted, if your merchant software and the WebSphere Commerce Payments both share the same realm, the double login problem (see "Writing a new WebSphere Commerce Payments realm" on page 50) can be removed by coding the merchant software links to pass authentication information through the WebSphere Commerce Payments user interface and into the realm code.

There are two ways that merchant software links can be coded to achieve single-signon:

1. by passing the user ID and password on the link
2. by passing a PMAUTHOBJECT string containing authentication information

Note that passing the user ID and password can often be seen as a security risk, particularly if this information is not protected by SSL. Both methods are discussed below in "Linking directly into the user interface" on page 54. You should choose whether you want to achieve single-signon and, if so, which method you want to support.

Realm scalability

The WebSphere Commerce Payments user interface will scale to support very large user registries, providing that the realm is also capable of scaling. When WebSphere Commerce Payments receives a QueryUsers command, the realm's getUserNames() method is called to get the users from the registry. If an inexperienced user queries WebSphere Commerce Payments for all users in a very large realm, it is possible that, a) the query may take a very long time, and b) the

WebSphere Commerce Payments JVM may run out of memory while processing the command. There are two methods that realms can make use of to control queries for very large numbers of users:

1. The first method is for the realm to limit the number of users it returns to WebSphere Commerce Payments. A UserList object is provided that is basically a Vector of user names along with a count of the total number of users that matched the query. You should decide whether or not you want to limit the number of users returned in response to a call to the getUserNames() method.
2. The second method is to make use of the userFilter passed via the QueryUsers API to the realm's getUserNames() method. The user filter should be used to search for a subset of users in the realm. However, its exact meaning is determined by the realm. The realms provided with WebSphere Commerce Payments all use this filter as a case-insensitive substring (that is, getUserNames() will return only user names that contain the filter string). The filter is made available on the WebSphere Commerce Payments user interface on the User Search window. You should decide whether you want to follow WebSphere Commerce Payments's convention of using this filter as a substring to search for, or to define this filter in another way.

Realm case-sensitivity

By default all WebSphere Commerce Payments realms are case-sensitive (that is, the user ID `admin` is different from the user ID `Admin`. If you want to provide a case-*insensitive* realm, you should override PaymentServletRealm's isCaseSensitive() method and return false.

Realm implementation

To write a realm, you must create a subclass of `PaymentServletRealm` and provide implementations for the methods listed below. You can find the Javadoc for the `PaymentServletRealm` class and the other classes it uses in the `docs/realmapi` directory where WebSphere Commerce Payments is installed.

- getRealmName()
- init()
- getAuthenticatedUser()
- isUserInRealm()
- getUserNames()

String getRealmName()

You should override this method to return a String that identifies the realm to the Payments administrator. This string is returned on the QueryPaymentServer API response and displayed both on the Basic settings screen and in the WebSphere Commerce Payments trace files to identify traced string originating from the realm. For more details, see "Tracing" on page 53.

void init(Properties properties) throws RealmException

This method is called when the PaymentServlet is first started. WebSphere Commerce Payments passes a Properties object which contains all the realm settings from the `<realmname>.properties` file. You should use these settings to initialize your realm. If settings are missing or invalid, you can choose either to use default settings or to go through a RealmException.

If you throw RealmException, the API response will contain a primary return code of 62 and a secondary return code of 1068 indicating to the calling program that the realm could not be initialized. You should ensure that you use the trace facility so that the exact problem can be diagnosed and corrected.

String getAuthenticatedUser(HttpServletRequest request) throws RealmException

For each API command received, WebSphere Commerce Payments will call the realm's getAuthenticatedUser() method to check that the user is a member of the realm and has provided valid authentication information. You can make use of the PaymentServletRealm's getFromHTTPAuthorizationString() and getPMAuthenticationString() convenience methods to pull authentication information from the HTTP request. This method must first determine the user's identity, and second determine whether the user is authenticated. If authenticated, the method should return the user name; otherwise, it should return null.

If you throw RealmException, the API response will contain a primary return code of 62 and a secondary return code of 1069 indicating to the calling program that a realm error occurred. You should ensure that you use the trace facility so that the exact problem can be diagnosed and corrected.

boolean isUserInRealm(String userName, String userFilter) throws RealmException

This method is used by WebSphere Commerce Payments when a user uses the QueryUsers API. The user filter must be null if there is no filtering. If this field is not null, you should implement your filtered search mechanism in this method.

WebSphere Commerce Payments ensures that each user it returns on the QueryUsers response is a valid member of the realm. To do this, it uses the realm's isUserInRealm() method. You should return true if the user is a valid member of the realm and any optional following criteria is met. Otherwise return false.

If you throw RealmException, the API response will contain a primary return code of 62 and a secondary return code of 1069 indicating to the calling program that a realm error occurred. You should ensure that you use the trace facility so that the exact problem can be diagnosed and corrected.

UserList getUserNames(String userFilter) throws RealmException

In response to a QueryUsers API call, WebSphere Commerce Payments can also invoke the realm's getUserNames() method to get a list of users in the realm that match the userFilter argument. You should implement your filtered search mechanism in this method.

If you throw RealmException, the API response will contain a primary return code of 62 and a secondary return code of 1069 indicating to the calling program that a realm error occurred. You should ensure that you use the trace facility so that the exact problem can be diagnosed and corrected.

Tracing

You can enable realm tracing via the Trace Settings screen in the WebSphere Commerce Payments User Interface. Select the check box underneath the 'Realm' column for WebSphere Commerce Payments and click Update to turn realm tracing on. Deselect the check box to turn it off.

For Windows and UNIX platforms:

- Trace output is sent to the `PMTrace1.log`and `PMTrace2.log` files which can be formatted with the provided **FormatTrace** command file. The realm trace output lines can be identified in the formatted log file by the prefix *REALM*.

For iSeries:

- Trace output is sent to the PMTrace1.logand PMTrace2.log files which can be formatted with the **Dump Payment Trace (DMPPYMTRC) CL** command, specifying *SERVLET for the trace type. The trace can also be formatted through the WebSphere Commerce Payments task page Service link.

The realm class can trace any information it chooses by using the following code segment:

```
if (isTracing()) trace("Hello World!");
```

For performance reasons, it is good practice to include the test for isTracing() since this will avoid making the Java run-time code evaluate the argument of the trace() method when tracing is disabled.

Linking directly into the user interface

In your Web pages you can provide links that bypass the login window and link directly into the user interface. During installation, a sample HTML file called **SampleSingleSignon.html** was installed in the **samples** directory where WebSphere Commerce Payments is installed. Using various methods of authentication, this sample shows you how to provide a link or button into one of WebSphere Commerce Payments's windows.

Authentication by the user interface

Authentication by the user interface can be performed in two ways. The two types of field data used for authentication are:

- f_userid=<userid> and f_password=<password>
- f_pmauthobject=<bytearray of authentication data>

The field data information that you provide to the user interface is passed to the PaymentServlet on all WebSphere Commerce Payments commands performed by the user interface. The realm gets the authentication information from the HTTP request it receives from the PaymentServlet.

The user interface servlet performs WebSphere Commerce Payments user interface functions only for authenticated users with proper authorization. To determine whether or not a user is authenticated, the user interface servlet needs authentication data. There are multiple sources where the user interface can find authentication data. Below is a list of sources and the order in which the user interface servlet accesses them once an HTTP POST is received from a user:

Note: The user interface servlet uses the first authentication data found.

1. If the user has previously logged in during this session, the authentication data saved when the user logged in originally is used on WebSphere Commerce Payments commands sent to the PaymentServlet.
2. If a WebSphere Commerce Payments Authentication Object is passed in the FORM data (using the f_pmauthobject keyword), the authentication data is used on WebSphere Commerce Payments commands sent to the PaymentServlet.
3. If a username/password is passed in the FORM data, (f_userid and f_password fields) the authentication data is used on WebSphere Commerce Payments commands sent to the PaymentServlet.
4. If a username/password is passed in through the Authorization header of the HTTP POST to the user interface servlet, the authentication data is used on WebSphere Commerce Payments commands sent to the PaymentServlet.

Once authentication data is found using the precedence order above, it is supplied on any commands sent to the PaymentServlet for the requested user interface function.

When an authentication failure occurs while linking directly to one of WebSphere Commerce Payments's user interface Web pages, the signon Web page appears and displays a login prompt. An authentication failure occurs if the PMAUTHOBJECT value is considered not valid according to the realm.

Testing

To test your realm, you will need to:

- Compile the realm
- Switch WebSphere Commerce Payments to use the realm
- Use the WebSphere Commerce Payments user interface and other programs to test the realm

The WebSphere Commerce Payments realm support classes are contained in `eTillClasses.zip`. When compiling the realm, you should remember to include this file and your realm class in your classpath.

Note that information on switching to use the new realm you created can be found in "How to deploy the new realm" on page 56.

SampleRealm

A sample realm class is provided with WebSphere Commerce Payments. This realm uses a database table called USERTABLE to hold a list of users along with their passwords. Since this is only a sample realm, all information is held unencrypted in the database. The first time SampleRealm is used, it will automatically create USERTABLE with three users (admin, fred and bob - all with the same passwords as user names).

You can find SampleRealm in the samples directory where you installed WebSphere Commerce Payments. To switch to using `SampleRealm`, edit the PaymentServlet init parameters via the WebSphere Application Server administrative console and set the wpm.RealmClass setting to *SampleRealm* - this is the fully qualified class name. Also, create a file called `SampleRealm.properties` in the WebSphere Commerce Payments installation directory and add the following content::

```
SampleRealmJDBCURL=<<the URL of the database that contains the USERTABLE>> --
for example, jdbc:db2:sampledb
SampleRealmDBDriver=<<the JDBC driver name>> --
for example, COM.ibm.db2.jdbc.app.DB2Driver
SampleRealmDBUserid=<<the userid that can be used to read USERTABLE>>
SampleRealmDBPassword=<<the password for SampleRealmDBUserid>>
```

When you restart your Webserver and WebSphere Application Server, this class will be enabled.

Note: The SampleRealm supports authentication via both the user ID/password combination and the PMAUTHOBJECT. Examples of how you can enable merchant software to implement single-signon can be found in an HTML file called `SampleSingleSignon.html` which can be found in the `samples` directory where you installed WebSphere Commerce Payments. To use `SampleSingleSignon.html`, you should copy this file to your Webserver's

publish directory and point a Web browser at
http://<<hostname>>/webapp/PaymentManager/SampleSingleSignon.html.

How to deploy the new realm

Installing the new realm is the responsibility of the realm writer. This section describes the steps necessary to deploy your new WebSphere Commerce Payments realm.

The Payment Servlet init parameter, wpm.RealmClass specifies which realm is to be used by WebSphere Commerce Payments. The value can be changed by opening the WebSphere Application Server administrative console. To manually configure your system to using your new realm, complete the following:

1. Open the WebSphere Application Server administrative console.
2. Expand **WebSphere Administrative Domain**.
3. Expand **Nodes**.
4. Expand the host name for the system where WebSphere Commerce Payments is installed.
5. Click **WebSphere Commerce Payments**. If you are using an iSeries system, click **WPM** *<instance>* **WebSphere Commerce Payments**, where *<instance>* is the name of the WebSphere Commerce Payments instance.
6. Select the **JVM Settings** tab page. In the **System Properties** box, add the realm, `<realmname>`.
7. Add the parameter value **com.ibm.commerce.payment.realm.<realmname>**.
8. Click **Apply**.
9. Stop and restart the WebSphere Commerce Payments Application Server in the WebSphere Application Server administrative console for your changes to take effect.

If the realm you are using needs additional configuration, then these realm settings will be specified in a file called `<realmname>.properties` in the WebSphere Commerce Payments installation directory.

For example, to configure WebSphere Commerce Payments to use the PSDefaultRealm, change the wpm.RealmClass Payment Servlet init parameter to be `com.ibm.etill.framework.payserverapi.PSDefaultRealm` and ensure that there is a file called `PSDefaultRealm.properties` in the WebSphere Commerce Payments installation directory that contains the line `RealmFile=<location of realm file>`.

Ensure that the realm class is in the `WebSphere Application Server` classpath. When the PaymentServlet is next initialized, your realm class will be loaded and start being used.

After updating the `wpm.RealmClass` setting and the realm-specific properties in the `<realmclass>.properties` file, you must restart your WebSphere Application Server for these changes to take effect.

For information on restarting your Web server and WebSphere Application Server, see ″Maintaining WebSphere Commerce Payments Security″ in the *WebSphere Commerce Payments Administrator's Guide*.

Part 3. Programmer's Reference

Chapter 7. WebSphere Commerce Payments command reference

Parameters for the commands described here apply to the Framework only. Additional parameters for specific cassettes are discussed in the appropriate cassette supplement. Note that in most cases, WebSphere Commerce Payments does not check for duplicate parameters. If more than one instance of a parameter is specified, then the last instance will be used.

Clients send commands to the WebSphere Commerce Payments by using HTTP POST requests, containing lists of keyword-value pairs. This chapter presents:

- WebSphere Commerce Payments financial and administrative commands
- Command descriptions
- Listing of required and optional keywords
- Guidelines regarding payment commands and query commands

Each command contains the name OPERATION. The value of the OPERATION parameter specifies the requested procedure.

In addition to OPERATION, ETAPIVERSION specifies the version number of the API. ETAPIVERSION is also required on every command.

Other name-value pairs in each command are dependent on the value of the OPERATION. The name-value pairs required by the payment operations are listed in the following tables. Other general guidelines for the name-value pairs include:

- The keyword strings are case-insensitive.
- Do not use leading zeros for any integers in ASCII characters.

Query commands

The following general rules apply to all queries:

- Each query has a set of *search modifiers* and a set of *operational parameters*. The modifiers determine the search criteria and the operational parameters affect the behavior or output of the command.
- All of the financial queries return either a ″collection″ or a ″keyCollection″ of the fundamental object being queried. The determination of collection versus keyCollection is made by the setting of the KEYSONLY parameter.
- Some keywords may be specified multiple times to achieve a search for a set of order values (for example, STATE=batch_opening, STATE=batch_open, STATE=batch_closed). For parameters that do not support multiple instances, the WebSphere Commerce Payments will not return an error and makes no guarantee as to which value will be used.
- To control the query results size, applications may use the RETURNATMOST parameter. RETURNATMOST limits the number of objects or object identifiers returned for a given query, even if that number is less than the actual number of objects that match the query. The maximum number of objects that can be returned is ten thousand. For more information on query results, see “Process returned objects” on page 41.
- One new property is introduced to the PaymentServlet.properties file to specify the minimum role a user must have to be allowed to view sensitive data. For each query command, the framework will check the user’s role against that

minimum role and will set an indicator in the QueryRequest object to indicate whether sensitive data should be returned in full view or if it should be masked out. For more information about setting the `wpm.MinSensitiveAccessRole` parameter, refer to the *WebSphere Commerce Payments Administrator's Guide*

About

The ABOUT command is typically used in two ways:

- As a ping mechanism to check to see if the WebSphere Commerce Payments is running.
- To return version information on the WebSphere Commerce Payments and the installed cassettes, as well as the username running the command.

For more information on the structured response returned by the ABOUT command, see "Payment Server About" on page 121 and "Cassette About" on page 121.

A successful execution of an ABOUT command will return primary and secondary return codes of ″0″, ″0″.

The ABOUT command is the only command that can be run by a *non*-authenticated user. When this command is run by a *non*-authenticated user, the command returns only a primary and secondary return code.

Required keywords for About command

Required Keywords	Value
ETAPIVERSION	″3″ (Indicates WebSphere Commerce Payments Version 2.1.x, 2.2.x and 3.1.x)
OPERATION	ASCII character string ″About″

Optional keywords for About command

Optional Keywords	Value
DTDPATH	Path to the locally-stored DTD. The value of this parameter used in the XML document to specify the location of the external DTD. If this parameter is not specified, the complete DTD is returned as an internal DTD. The length of the DTDPath must be from 1 to 254 bytes.

AcceptPayment

Use the ACCEPTPAYMENT command to create Order objects when an electronic wallet is not used. In general, if the command is successful, the order will be placed in Ordered state. If the command fails, the order will not be created. Pass protocol specific data on this command; however, specifics depend on the cassette. Refer to the particular cassette supplement for details

During the processing of an AcceptPayment command, you can ensure that the cassette handles the Approval step separately from the Order creation step. Select the Asynchronous Auto Approve payment processing option to indicate that the approval is asynchronously scheduled to occur. Thus, the buyer does not have to wait for the approval to occur before receiving a response for the original purchase request.

When creating an order, you may want bto approve or deposit funds automatically. The APPROVEFLAG and DEPOSITFLAG keywords indicate whether or not a Payment object should be approved and deposited. Refer to the appropriate table below for additional keywords used if APPROVEFLAG or DEPOSITFLAG are specified.

Required keywords for AcceptPayment command

Required Keywords	**Value**
AMOUNT	A positive 32-bit integer in ASCII characters.
CURRENCY	Integer in ASCII characters. See Appendix B, Currency Codes, for a list of ISO currency codes.
ETAPIVERSION	″3″ (Indicates WebSphere Commerce Payments Version 2.1.x, 2.2.x and 3.1.x)
MERCHANTNUMBER	Integer in ASCII characters. Must be from 1 to 999999999.
OPERATION	ASCII character string ″AcceptPayment″
ORDERNUMBER	Integer in ASCII characters. Must be from 1 to 999999999.
PAYMENTTYPE	ASCII character string. Specifies the payment protocol being used. For example, OfflineCard..

Optional keywords for AcceptPayment command

Optional Keywords	**Value**
AMOUNTEXP10	Integer in ASCII characters. Indicates the number of decimal places to shift. Valid values are -10 to 10. for more information about this keyword, refer to “Using the AmountExp10 keyword” on page 62.
APPROVEFLAG	Integer in ASCII characters. Indicates whether the approvals should be attempted automatically. Default is 0. Supported values are: 0 - Indicates transaction should not be approved. 1 - Indicates transaction should be approved automatically. 2 - Indicates transaction should be approved asynchronously.
DTDPATH	Path to the locally stored DTD. The value of this parameter is used in the XML document to specify the location of the external DTD. If this parameter is not specified, the complete DTD is returned as an internal DTD. The length of the DTDPath must be from 1 to 254 bytes.
ORDERDATA1	Auxiliary data supplied by the user, specified as an ASCII character string between 1 and 254 bytes in length.
ORDERDATA2	Auxiliary data supplied by the user, specified as a UTF-8 string between 1 and 254 bytes in length.
ORDERDATA3	Auxiliary data supplied by the user, specified as a UTF-8 string between 1 and 254 bytes in length.
ORDERDATA4	Auxiliary data supplied by the user, specified as a binary string between 1 and 254 bytes in length.
ORDERDATA5	Auxiliary data supplied by the user, specified as a binary string with an arbitrary length.
ORDERURL	URL containing order details.

Optional keywords for AcceptPayment command

TRANSACTIONID	Transaction identifier supplied by the user, specified as an ASCII character string between 1 and 128 bytes in length.

The following tables list the required and optional keywords for APPROVEFLAG=1 or 2.

Required keywords if APPROVEFLAG is set to 1 or 2.

Required Keywords	Value
PAYMENTAMOUNT	A 32-bit positive integer in ASCII characters.
PAYMENTNUMBER	Integer in ASCII characters. Must be from 1 to 999999999.

Optional keywords if APPROVEFLAG is set to 1or 2.

Optional Keywords	Value
DEPOSITFLAG	Boolean value in ASCII characters. Indicates whether the deposit should be attempted automatically. This flag is only valid if APPROVE=1 (order is automatically approved). Supported values are: 0 - Funds should not be automatically deposited 1 - Funds should be automatically deposited.

If DEPOSITFLAG=1, then the following keyword is optional:

Optional keyword if DEPOSITFLAG is set to 1.

Optional Keywords	Value
BATCHNUMBER	Identifies the batch under which this payment will be processed. Must be from 1 to 999999999.

Using the AmountExp10 keyword

All amount values are expressed as an amount with currency and exponent. For example, $5.00 USD (U.S. Dollars) is expressed with Amount=500, Currency=840 (the ISO currency code for USD), and AmountExp10 =-2.

All current ISO currencies have exactly one valid exponent value, so the exponent can be inferred from the currency. The WebSphere Commerce Payments maintains a mapping table from currencies to exponents as shown in Appendix B, Currency Codes. During order creation, (that is, on RECEIVEPAYMENT or ACCEPTPAYMENT commands), merchant software must always specify both AMOUNT and CURRENCY keywords. If the currency specified is a known currency in the ISO table, the corresponding exponent will be used. If the currency specified is not known (that is, it is not present in the ISO table), then an additional parameter (AMOUNTEXP10) will be needed to specify the exponent. The existence of the AMOUNTEXP10 parameter allows for flexibility in supporting future currencies.

AMOUNTEXP10 Specified on API	CURRENCY Present in Mapping Table

True	True If the exponent passed in on the AMOUNTEXP10 parameter is the same as the one in the mapping table, then the exponent is used. If the exponent passed in differs from the one in the table, then a parameter error occurs.
True	False The exponent passed in on the AMOUNTEXP10 parameter is used.
False	True The exponent derived from the mapping table is used.
False	False A ″parameter not found″ error occurs.

Approve

The APPROVE command is used by the merchant to ask the financial system if the buyer should be allowed to make the purchase. For example, for a credit card system, this command would result in a credit card authorization.

The APPROVE command creates a new Payment object for an existing order. This command is legal when the order is in Ordered or Refundable state. If successful, the payment will be in either Approved, Deposited, or Closed state if DEPOSITFLAG is set to 1. If unsuccessful, the payment will be in Declined state.

When approving a payment, you may want to make a deposit automatically. The DEPOSITFLAG keyword indicates that a Payment object should be deposited. Refer to the appropriate table below for additional keywords, if DEPOSITFLAG is set to 1.

Required keywords for Approve command

Required Keywords	**Value**
AMOUNT	A positive 32-bit integer in ASCII characters.
ETAPIVERSION	″3″ (Indicates WebSphere Commerce Payments Version 2.1.x, 2.2.x and 3.1.x)
MERCHANTNUMBER	Integer in ASCII characters. Must be from 1 to 999999999.
OPERATION	ASCII character string ″Approve″
ORDERNUMBER	Integer in ASCII characters. Must be from 1 to 999999999.
PAYMENTNUMBER	Integer in ASCII characters. Must be from 1 to 999999999.

Optional keywords for Approve command

Optional Keywords	**Value**
DEPOSITFLAG	Indicates whether the approved payment should be deposited automatically. Default is 0. Supported values are: 0 - Funds should not be automatically deposited. 1 - Funds should be automatically deposited.

Optional keywords for Approve command

DTDPATH	Path to the locally stored DTD. The value of this parameter is used in the XML document to specify the location of the external DTD. If this parameter is not specified, the complete DTD is returned as an internal DTD. The length of the DTDPath must be from 1 to 254 bytes.

The following keyword is optional if DEPOSITFLAG=1.

Optional keywords if DEPOSITFLAG is set to 1.

Optional Keywords	Value
BATCHNUMBER	Identifies the batch under which this payment will be processed. A numeric string of up to nine characters. Must be from 1 to 999999999.

ApproveReversal

An ApproveReversal command modifies the approved amount of a payment. For example, if a payment enters the ApprovalExpired state, then you can use the ApproveReversal command either to get a new approval or to void the payment. ApproveReversal is valid for payments in the Approved state. If the ApproveReversal is successful, and the amount specified is ″0,″ then the payment moves to Void state. If the amount specified is not ″0,″ then the payment stays in Approved state and the approved amount is modified.

Required keywords for ApproveReversal command

Required Keywords	Value
AMOUNT	Must be positive 32-bit integer in ASCII characters.
ETAPIVERSION	″3″ (Indicates WebSphere Commerce Payments Version 2.1.x, 2.2.x and 3.1.x)
MERCHANTNUMBER	Integer in ASCII characters. Must be from 1 to 999999999.
OPERATION	ASCII character string ″ApproveReversal″
ORDERNUMBER	Integer in ASCII characters. Must be from 1 to 999999999.
PAYMENTNUMBER	Integer in ASCII characters. Must be from 1 to 999999999.

Optional keywords for ApproveReversal command

Optional Keywords	Value
DTDPATH	Path to the locally stored DTD. The value of this parameter is used in the XML document to specify the location of the external DTD. If this parameter is not specified, the complete DTD is returned as an internal DTD. The length of the DTDPath must be from 1 to 254 bytes.

BatchClose

The BATCHCLOSE command closes a batch and moves the Batch object into Closed state. All Payment and Credit objects associated with this batch move to Closed state as well. This command is only permissible if:

- The batch is in Open state
- The account allows the merchant to close the batch

- The merchant control attribute is set to true

Required keywords for BatchClose command

Required Keywords	Value
BATCHNUMBER	A numeric string of up to nine characters. Must be from 1 to 999999999.
ETAPIVERSION	″3″ (Indicates WebSphere Commerce Payments Version 2.1.x, 2.2.x and 3.1.x)
MERCHANTNUMBER	Integer in ASCII characters. Must be from 1 to 999999999.
OPERATION	ASCII character string ″BatchClose″

Optional keywords for BatchClose command

Optional Keywords	Value
DTDPATH	Path to the locally stored DTD. The value of this parameter is used in the XML document to specify the location of the external DTD. If this parameter is not specified, the complete DTD is returned as an internal DTD. The length of the DTDPath must be from 1 to 254 bytes.
FORCE	Valid values are ″0″ and ″1″. A value of ″1″ indicates a local close should be performed even if the financial operation fails.

BatchOpen

The BATCHOPEN command creates a Batch object and, if successful, puts the batch into Open state. This command is only permissible if the account allows merchants to open batches.

Note: In a scenario where there is one merchant (123456789), with two accounts (acct#1, acct#2), if a BatchOpen is issued with acct#1, batch#1, the batch will open. When a BatchOpen is sent with acct#2, batch#1, the BatchOpen will fail and the following message is displayed:

```
Tue Jun22 13:04:31 EDT 1999 CEPFW0715: Batch ID 299 already exists for
Merchant 123456789 and account 2.
```

The second test will fail because only one batch with a given BatchNumber can be in the system at any one time.

Required keywords for BatchOpen command

Required Keywords	Value
OPERATION	ASCII character string ″BatchOpen″
ACCOUNTNUMBER	Integer in ASCII characters. This value is a unique ID that indicates the acquirer to the merchant. The value must match the WebSphere Commerce Payments configured AccountNumber value. Must be from 1 to 999999999.
BATCHNUMBER	A numeric string of up to nine characters. Must be from 1 to 999999999.

Required keywords for BatchOpen command

ETAPIVERSION	″3″ (Indicates WebSphere Commerce Payments Version 2.1.x, 2.2.x and 3.1.x)
MERCHANTNUMBER	Integer in ASCII characters. Must be from 1 to 999999999.
PAYMENTTYPE	ASCII character string that identifies the payment type.

Optional keyword for BatchOpen command

Optional Keywords	**Value**
DTDPATH	Path to the locally stored DTD. The value of this parameter is used in the XML document to specify the location of the external DTD. If this parameter is not specified, the complete DTD is returned as an internal DTD. The length of the DTDPath must be from 1 to 254 bytes.

BatchPurge

The BATCHPURGE command clears out a batch and returns the Batch object to Open state. All Payment and Credit objects associated with this batch are removed from the batch, with Payment objects returned to Approved state and Credit objects returned to Void state. This command is only permissible if the PurgeAllowed attribute is set to true.

Required keywords for BatchPurge command

Required Keywords	**Value**
BATCHNUMBER	A numeric string of up to nine characters. Must be from 1 to 999999999.
ETAPIVERSION	″3″ (Indicates WebSphere Commerce Payments Version 2.1.x, 2.2.x and 3.1.x)
MERCHANTNUMBER	Integer in ASCII characters. Must be from 1 to 999999999.
OPERATION	ASCII character string ″BatchPurge″

Optional keywords for BatchPurge command

Optional Keywords	**Value**
DTDPATH	Path to the locally stored DTD. The value of this parameter is used in the XML document to specify the location of the external DTD. If this parameter is not specified, the complete DTD is returned as an internal DTD. The length of the DTDPath must be from 1 to 254 bytes.

CancelOrder

The CANCELORDER command moves an Order into Canceled state. You can invoke the CancelOrder command for an Order that satisfies the following criteria:

- It has no Payments or Credits associated with it, OR

- Any associated Payments or Credits are in their respective Reset, Void, ApprovalExpired or Declined state.

Once an Order is in Canceled state, no operations are legal except for CancelOrder. If the optional parameter, DELETEORDER, is set to ″1,″ then the Order will be pruned. All related Payments and Credits will also be deleted; cassette-specific objects will be deleted as well.

Required keywords for CancelOrder command

Required Keywords	Value
ETAPIVERSION	″3″ (Indicates WebSphere Commerce Payments Version 2.1.x, 2.2.x and 3.1.x)
MERCHANTNUMBER	Integer in ASCII characters. Must be from 1 to 999999999.
OPERATION	ASCII character string ″CancelOrder″
ORDERNUMBER	Integer in ASCII characters. Must be form 1 to 999999999.

Optional keywords for CancelOrder command

Optional Keywords	Value
DELETEORDER	Indicates that the order and all ancillary objects should be deleted. Default is ″0″. Supported values are: 0-Objects should not be deleted 1-Objects should be deleted
DTDPATH	Path to the locally stored DTD. The value of this parameter is used in the XML document to specify the location of the external DTD. If this parameter is not specified, the complete DTD is returned as an internal DTD. The length of the DTDPath must be from 1 to 254 bytes.

CassetteControl

The CASSETTECONTROL command is used to perform cassette-specific functions that do not correspond to any generic commands. CASSETTECONTROL is not interpreted by the Framework, but is passed down to the cassette.

Refer to the appropriate cassette supplement for details on how this command is used.

Required keywords for CassetteControl command

Required Keywords	Value
CASSETTECOMMAND	Command name in ASCII characters. Maximum length is 1000 bytes.
ETAPIVERSION	″3″ (Indicates WebSphere Commerce Payments Version 2.1.x, 2.2.x and 3.1.x).
OPERATION	ASCII character string ″CassetteControl″.
PAYMENTTYPE	ASCII character string. Specifies the payment protocol being used.

Optional keywords for CassetteControl command

Optional Keywords	Value
DTDPATH	Path to the locally-stored DTD. The value of this parameter used in the XML document to specify the location of the external DTD. If this parameter is not specified, the complete DTD is returned as an internal DTD. The length of the DTDPath must be from 1 to 254 bytes.

CloseOrder

The CLOSEORDER command moves an Order into Closed state. You can invoke the CLOSEORDER command for an Order that satisfies the following criteria:

- It has at least one Payment or Credit associated with it, AND
- All of the Payments and Credits associated with the Order are in their respective Closed state.

Once an Order is in Closed state, no operations are legal on it except for CancelOrder. If the optional parameter DELETEORDER is set to ″1″, then the database will be pruned, so you can call CloseOrder on an Order in Closed state. Payments and Credits must be closed.

Required keywords for CloseOrder

Required Keywords	Value
ETAPIVERSION	″3″ (Indicates WebSphere Commerce Payments Version 2.1.x, 2.2.x and 3.1.x).
MERCHANTNUMBER	Integer in ASCII characters. Must be from 1 to 999999999.
OPERATION	ASCII character string ″CloseOrder″.
ORDERNUMBER	Integer in ASCII characters. Must be from 1 to 999999999.

Optional keywords for CloseOrder command

Optional Keywords	Value
DELETEORDER	Indicates that the order and all ancillary objects should be deleted. Default is 0. Supported values are: 0-Order and all ancillary objects should not be deleted 1-Order and all ancillary objects should be deleted
DTDPATH	Path to the locally stored DTD. The value of this parameter is used in the XML document to specify the location of the external DTD. If this parameter is not specified, the complete DTD is returned as an internal DTD. The length of the DTDPath must be from 1 to 254 bytes.

CreateAccount

The CREATEACCOUNT command creates an Account object for the specified Payment System object.

Required keywords for CreateAccount command

Required Keywords	Value
ACCOUNTNUMBER	Integer in ASCII characters. Must be from 1 to 999999999. Specifies an identifier for the new Account.
CASSETTENAME	ASCII character string from 1 to 64 bytes. Specifies an identifier for the new account.
ETAPIVERSION	″3″ (Indicates WebSphere Commerce Payments Version 2.1.x, 2.2.x and 3.1.x)
MERCHANTNUMBER	Integer in ASCII characters. Must be from 1 to 999999999. Specifies an identifier for the new Account.
OPERATION	ASCII character string ″CreateAccount″

Optional keywords for CreateAccount command

Optional Keywords	Value
ACCOUNTTITLE	UTF-8 string that is either null or from 1 to 254 bytes. If present, the value passed in will replace the AccountTitle specified Account object.
DTDPATH	Path to the locally-stored DTD. The value of this parameter is used in the XML document to specify the location of the external DTD. If this parameter is not specified, the complete DTD is returned as an internal DTD. The length of the DTDPath must be from 1 to 254 bytes.
ENABLED	ASCII character string ″0″ or ″1,″ where ″0″ and ″1″ denote false and true, respectively. If present, the value passed in will replace the Enabled field of the Account object. Indicates whether the Account object should be active.
FINANCIALINSTITUTION	UTF-8 string that is either null or from 1 to 254 bytes. If present, the value passed in will replace the Financial Institution specified Account object.
APAPPROVEFLAG	Approve flag for AcceptPayment. ASCII character string ″0″, ″1″, or ″2″. Default is ″0″ ″0″ indicates transaction should not be approved. ″1″ indicates transaction should be approved automatically. ″2″ indicates transaction should be approved asynchronously.

Optional keywords for CreateAccount command

RPAPPROVEFLAG	Approve flag for ReceivePayment. ASCII character string ″0″, ″1″, or ″2″. Default is ″0″ ″0″ indicates transaction should not be approved. ″1″ indicates transaction should be approved automatically. ″2″ indicates transaction should be approved asynchronously.
APDEPOSITFLAG	ASCII character string ″0″ or ″1″, where ″0″ and ″1″ denote false and true, respectively. Only specified if APAPPROVEFLAG is defined and not set to 0. Otherwise PRC_INVALID_PARAMETER_ COMBINATION_, RC_AP_DEPOSITFLAG will be returned.
RPDEPOSITFLAG	ASCII character string ″0″ or ″1″, where ″0″ and ″1″ denote false and true, respectively. Only specified if RPAPPROVEFLAG is defined and not set to 0.Otherwise PRC_INVALID_PARAMETER_ COMBINATION_, RC_RP_DEPOSITFLAG will be returned.
APPROVALEXPIRATION	Integer value that indicates the number of days after a payment has been approved that the approval expires. This field supports configurable approval expiration where this setting controls whether a payment approval associated with the account will expire after the elapsed time. A value of 0 implies no expiration. When a payment approval expires, it will be placed in the ApprovalExpired state. **Note:** A cassette is allowed to cause payment approvals to expire independently of this setting, but this parameter allows the framework to detect payment approval expiration on behalf of the cassette. Add a cross reference to Payment States to see a description of the ApprovalExpired state.

Note: APAPPROVEFLAG AND RPAPPROVEFLAG values are superseded by the API Approve flag when the API Approve flag contains a non-zero, non-null value.

CreateMerchant

The CREATEMERCHANT command creates a Merchant object.

Required keywords for CreateMerchant command

Required Keywords	Value
ETAPIVERSION	″3″ (Indicates WebSphere Commerce Payments Version 2.1.x, 2.2.x and 3.1.x).
MERCHANTNUMBER	Integer in ASCII characters. Must be from 1 to 999999999. Specifies the identifier for the new Merchant object.
OPERATION	ASCII character string ″CreateMerchant″.

Optional keywords for CreateMerchant command

Optional Keywords	Value
DTDPATH	Path to the locally-stored DTD. The value of this parameter is used in the XML document to specify the location of the external DTD. If this parameter is not specified, the complete DTD is returned as an internal DTD. The length of the DTDPath must be from 1 to 254 bytes.
ENABLED	ASCII character string ″0″ or ″1,″ where ″0″ and ″1″ denote false and true, respectively. If present, the value passed in will replace the Enabled field of the Merchant object. Indicates whether the Merchant object should be active.
MERCHANTTITLE	UTF-8 string that is either null or from 1 to 128 bytes. If present; the value passed in will replace the MerchantTitle specified Merchant object.

CreateMerchantCassetteObject

The CREATEMERCHANTCASSETTEOBJECT command is used to create a cassette-specific object with the type specified in the OBJECTNAME keyword.

Refer to the appropriate cassette supplement for details on how this command is used.

Required keywords for CreateMerchantCassetteObject command

Required Keywords	Value
CASSETTENAME	ASCII character string from 1 to 64 bytes. Specifies an identifier for the MerchantCassette object.
ETAPIVERSION	″3″ (Indicates WebSphere Commerce Payments Version 2.1.x, 2.2.x and 3.1.x).
MERCHANTNUMBER	Integer in ASCII characters. Must be from 1 to 999999999. Specifies an identifier for MerchantCassette object.
OBJECTNAME	ASCII character string. Value specified by the cassette. Specifies an identifier for MerchantCassette object. Maximum length is 1000 bytes.
OPERATION	ASCII character string ″CreateMerchantCassetteObject″

Optional keywords for the CreateMerchantCassetteObject command.

Optional Keywords	Value
DTDPATH	Path to the locally-stored DTD. The value of this parameter is used in the XML document to specify the location of the external DTD. If this parameter is not specified, the complete DTD is returned as an internal DTD. The length of the DTDPath must be from 1 to 254 bytes.
ENABLED	ASCII character string ″0″ or ″1″, where ″0″ and ″1″ denote false and true, respectively. If present, the value passed in will replace the Enabled field of the MerchantCassette object. Indicates whether the MerchantCassette object should be enabled.

CreateMerEventListener

The CREATEMEREVENTLISTENER command creates a merchant event listener.

Required keywords for CreateMerEventListener command

Required Keywords	Value
ETAPIVERSION	″3″ (Indicates WebSphere Commerce Payments Version 2.1.x, 2.2.x and 3.1.x).
EVENTTYPE	Integer in ASCII characters that identifies the event type. Events have the following values: 1: State change event 2: Cassette-specific event
LISTENERURL	ASCII character string that identifies where the events show (for example, http://www.merchant.com/webapp/PaymentManager /eventReceiver888). If the port number is not specified, the default port number, 80, is used. A valid URL from 1 to 256 characters.
MERCHANTNUMBER	Integer in ASCII characters. Must be from 1 to 999999999.
OPERATION	ASCII character string ″CreateMerEventListener″.

Optional keywords for CreateMerEventListener command

Optional Keywords	Value
DTDPATH	Path to the locally-stored DTD. The value of this parameter is used in the XML document to specify the location of the external DTD. If this parameter is not specified, the complete DTD is returned as an internal DTD. The length of the DTDPath is from 1 to 254 bytes.
SOCKSHOST	Host name of the socks server. This parameter is required only for the event being sent through a socks server. Maximum length is 256 bytes.
SOCKSPORT	Port number of the socks server. This parameter is only used if SOCKSHOST is specified. The default is 1080. The value for a (nonnull) SocksPort parameter must be a positive 16-bit unsigned integer from 1 to 65535.

Required keywords if EventType is set to 2.

Required Keywords	Value
CASSETTENAME	ASCII character string up to 64 bytes that identifies the cassette name. Required for registering cassette events. No parameter limitations-must match an existing cassette or will fail.

CreatePaySystem

The CREATEPAYSYSTEM command creates a Payment System object for assigning the specified merchant permission to use the specified cassette.

Required keywords for CreatePaySystem command

Required Keywords	Value

Required keywords for CreatePaySystem command

CASSETTENAME	ASCII character string from 1 to 64 bytes. Specifies an identifier for the new PaySystem object.
ETAPIVERSION	″3″ (Indicates WebSphere Commerce Payments Version 2.1.x, 2.2.x and 3.1.x).
MERCHANTNUMBER	Integer in ASCII characters. Must be from 1 to 999999999. Specifies an identifier for the new PaySystem object.
OPERATION	ASCII character string ″CreatePaySystem″.

Optional keywords for CreatePaySystem command

Optional Keywords	Value
DTDPATH	Path to the locally-stored DTD. The value of this parameter is used in the XML document to specify the location of the external DTD. If this parameter is not specified, the complete DTD is returned as an internal DTD. The length of the DTDPath must be from 1 to 254 bytes.
ENABLED	ASCII character string ″0″ or ″1,″ where ″0″ and ″1″ denotes false and true, respectively. If present, the value passed in will replace the Enabled field of the PaySystem object. Indicates whether the PaySystem object should be active.

CreateSNMEventListener

The CREATESNMEVENTLISTENER command creates a system network management event listener.

Required keywords for CreateSNMEventListener command

Required Keywords	Value
ETAPIVERSION	″3″ (Indicates WebSphere Commerce Payments Version 2.1.x, 2.2.x and 3.1.x)
EVENTTYPE	″3″ (Identifies the SNM event type.) Other values reserved for future use.
LISTENERURL	ASCII character string that identifies where the events show (for example, http://www.merchant.com/webapp/PaymentManager /eventReceiver888). If the port number is not specified, the default port number, 80, is used. A valid URL from 1 to 256 characters.
OPERATION	ASCII character string ″CreateSNMEventListener″.

Optional keywords for CreateSNMEventListener command

Optional Keywords	Value
DTDPATH	Path to the locally-stored DTD. The value of this parameter is used in the XML document to specify the location of the external DTD. If this parameter is not specified, the complete DTD is returned as an internal DTD. The length of the DTDPath must be from 1 to 254 bytes.

Optional keywords for CreateSNMEventListener command

SOCKSHOST	Host name of the socks server. This parameter is required only for the event being sent through a socks server. Parameter values must be a valid integer (if specified). Maximum length is 256 bytes.
SOCKSPORT	Port number of the socks server. This parameter is only used if SOCKSHOST is specified. The default is 1080. The value for a (nonnull) SocksPort parameter must be a positive 16-bit unsigned integer from 1 to 65535.

CreateSystemCassetteObject

The CREATESYSTEMCASSETTEOBJECT command creates a cassette-specific object with the type specified in the OBJECTNAME keyword.

Refer to the appropriate cassette supplement for details on how this command is used.

Required keywords for CreateSystemCassetteObject command

Required Keywords	Value
CASSETTENAME	ASCII character string from 1 to 64 bytes. Specifies an identifier for the SystemCassette object.
ETAPIVERSION	″3″ (Indicates WebSphere Commerce Payments Version 2.1.x, 2.2.x and 3.1.x).
OBJECTNAME	ASCII character string. Value specified by the cassette. Specifies an identifier for the SystemCassette object.
OPERATION	ASCII character string ″CreateSystemCassetteObject″.

Optional keywords for CreateSystemCassetteObject command

Optional Keywords	Value
DTDPATH	Path to the locally-stored DTD. The value of this parameter is used in the XML document to specify the location of the external DTD. If this parameter is not specified, the complete DTD is returned as an internal DTD. The length of the DTDPath must be from 1 to 254 bytes.
ENABLED	ASCII character string ″0″ or ″1,″ where ″0″ and ″1″ denote false and true, respectively. If present, the value passed in will replace the Enabled field of the SystemCassette object. Indicates whether the SystemCassette object should be active.

DeleteAccount

The DELETEACCOUNT command deletes the specified Account object and all its subsidiary objects.

Required keywords for DeleteAccount command

Required Keywords	Value

Required keywords for DeleteAccount command

ACCOUNTNUMBER	Integer in ASCII characters. Must be from 1 to 999999999. In conjunction with MERCHANTNUMBER and CASSETTENAME, it uniquely identifies the target Account object for this command.
CASSETTENAME	ASCII character string from 1 to 64 bytes. In conjunction with MERCHANTNUMBER and ACCOUNTNUMBER, uniquely identifies the target Account object for this command.
ETAPIVERSION	″3″ (Indicates WebSphere Commerce Payments Version 2.1.x, 2.2.x and 3.1.x).
MERCHANTNUMBER	Integer in ASCII characters. Must be from 1 to 999999999. In conjunction with CASSETTENAME and ACCOUNTNUMBER, it uniquely identifies the target Account object for this command.
OPERATION	ASCII character string ″DeleteAccount″.

Optional keyword for DeleteAccount command.

Optional Keyword	Value
DTDPATH	Path to the locally-stored DTD. The value of this parameter is used in the XML document to specify the location of the external DTD. If this parameter is not specified, the complete DTD is returned as an internal DTD. The length of the DTDPath must be from 1 to 254 bytes.

DeleteBatch

The DELETEBATCH command prunes the specified batch from the database tables. The DELETEBATCH command is legal only when a batch is in Closed state.

Required keywords for DeleteBatch command

Required Keywords	Value
BATCHNUMBER	Integer in ASCII characters. Identifies the number of the batch which this payment is assigned. Must be from 1 to 999999999.
ETAPIVERSION	″3″ (Indicates WebSphere Commerce Payments Version 2.1.x, 2.2.x and 3.1.x).
MERCHANTNUMBER	Integer in ASCII characters. Must be from 1 to 999999999.
OPERATION	ASCII character string ″DeleteBatch″.

Optional keyword for DeleteBatch command

Optional Keyword	Value
DTDPATH	Path to the locally stored DTD. The value of this parameter is used in the XML document to specify the location of the external DTD. If this parameter is not specified, the complete DTD is returned as an internal DTD. The length of the DTDPath must be from 1 to 254 bytes.

DeleteMerchant

The DELETEMERCHANT command deletes the specified Merchant object and all its subsidiary objects.

Required keywords for DeleteMerchant command

Required Keywords	Value
ETAPIVERSION	″3″ (Indicates WebSphere Commerce Payments Version 2.1.x, 2.2.x and 3.1.x).
MERCHANTNUMBER	Integer in ASCII characters. Must be from 1 to 999999999. Use the target Merchant object for this command.
OPERATION	ASCII character string ″DeleteMerchant″.

Optional keyword for DeleteMerchant command

Optional Keyword	Value
DTDPATH	Path to the locally-stored DTD. The value of this parameter is used in the XML document to specify the location of the external DTD. If this parameter is not specified, the complete DTD is returned as an internal DTD. The length of the DTDPath must be from 1 to 254 bytes.

DeleteMerchantCassetteObject

The DELETEMERCHANTCASSETTEOBJECT command deletes the cassette object with the type specified by the object name.

Refer to the appropriate cassette supplement for details on how this command is used.

Required keywords for DeleteMerchantCassetteObject command

Required Keywords	Value
CASSETTENAME	ASCII character string from 1 to 64 bytes. In conjunction with MERCHANTNUMBER, OBJECTNAME and protocol data parameters, uniquely identifies the target MerchantCassette for this command.
ETAPIVERSION	″3″ (Indicates WebSphere Commerce Payments Version 2.1.x, 2.2.x and 3.1.x).
MERCHANTNUMBER	Integer in ASCII characters. Must be from 1 to 999999999. In conjunction with CASSETTENAME, OBJECTNAME and data parameters, uniquely identifies the target MerchantCassette for this command.
OBJECTNAME	ASCII character string value specified by the cassette. In conjunction with CASSETTENAME, MERCHANTNUMBER protocol data parameters, uniquely identifies the target MerchantCassette object for this command. The maximum length is 1000 bytes.
OPERATION	ASCII character string ″DeleteMerchantCassetteObject″

Optional keyword for DeleteMerchantCassetteObject command.

Optional Keyword	Value

Optional keyword for DeleteMerchantCassetteObject command.

DTDPATH	Path to the locally-stored DTD. The value of this parameter is used in the XML document to specify the location of the external DTD. If this parameter is not specified, the complete DTD is returned as an internal DTD. The length of the DTDPath must be from 1 to 254 bytes.

DeleteMerEventListener

The DELETEMEREVENTLISTENER command deletes the MerEventListener object.

Required keywords for DeleteMerEventListener command

Required Keywords	Value
CASSETTENAME	ASCII character string up to 64 bytes that identifies the cassette name. Required for registering cassette events. No parameter limitations-must match an existing cassette or will fail.
ETAPIVERSION	'3″ (Indicates WebSphere Commerce Payments Version 2.1.x, 2.2.x and 3.1.x).
EVENTTYPE	Integer in ASCII characters that identifies the event type. Events have the following values: 1: State change event 2: Cassette-specific event
LISTENERURL	ASCII character string that identifies where the events show (for example, http://www.merchant.com/webapp/PaymentManager /eventReceiver888). If the port number is not specified, the default port number, 80, is used. No parameter limitations.
MERCHANTNUMBER	Integer in ASCII characters. Must be from 1 to 999999999.
OPERATION	ASCII character string ″DeleteMerEventListener″.

Optional keywords for DeleteMerEventListener command

Optional Keywords	Value
DTDPATH	Path to the locally-stored DTD. The value of this parameter is used in the XML document to specify the location of the external DTD. If this parameter is not specified, the complete DTD is returned as an internal DTD. The length of the DTDPath must be from 1 to 254 bytes.

DeletePaySystem

The DELETEPAYSYSTEM command deletes the specified Payment System object and all its subsidiary objects.

Required keywords for DeletePaySystem command

Required Keywords	Value
CASSETTENAME	ASCII character string from 1 to 64 bytes. In conjunction with MERCHANTNUMBER, uniquely identifies the target MerchantCassetteSettings object for this command.

Required keywords for DeletePaySystem command

ETAPIVERSION	″3″ (Indicates WebSphere Commerce Payments Version 2.1.x, 2.2.x and 3.1.x).
MERCHANTNUMBER	Integer in ASCII characters. Must be from 1 to 999999999. In conjunction with CASSETTENAME, uniquely identifies the MerchantCassetteSettings object for this command.
OPERATION	ASCII character string ″DeletePaySystem″.

Optional keyword for DeletePaySystem command

Optional Keyword	Value
DTDPATH	Path to the locally-stored DTD. The value of this parameter is used in the XML document to specify the location of the external DTD. If this parameter is not specified, the complete DTD is returned as an internal DTD. The length of the DTDPath must be from 1 to 254 bytes.

DeleteSNMEventListener

The DELETESNMEVENTLISTENER command deletes the specified system network management event listener.

Required keywords for DeleteSNMEventListener command

Required Keywords	Value
ETAPIVERSION	″3″ (Indicates WebSphere Commerce Payments Version 2.1.x, 2.2.x and 3.1.x).
EVENTTYPE	″3″ (Identifies the SNM event type. Other values reserved for future use.)
LISTENERURL	ASCII character string that identifies where the events show (for example, http://www.merchant.com/webapp/PaymentManager /eventReceiver888). If the port number is not specified, the default port number, 80, is used. A valid URL from 1 to 256 characters.
OPERATION	ASCII character string ″DeleteSNMEventListener″.

Optional keyword for DeleteSNMEventListener command

Optional Keywords	Value
DTDPATH	Path to the locally-stored DTD. The value of this parameter is used in the XML document to specify the location of the external DTD. If this parameter is not specified, the complete DTD is returned as an internal DTD. The length of the DTDPath must be from 1 to 254 bytes.

DeleteSystemCassetteObject

The DELETESYSTEMCASSETTEOBJECT command deletes the Cassette object with type specified by object name.

Refer to the appropriate cassette supplement for details on how this command is used.

Required keywords for DeleteSystemCassetteObject command

Required Keywords	Value
CASSETTENAME	ASCII character string from 1 to 64 bytes. In conjunction with OBJECTNAME and protocol data parameters, uniquely identifies the target SystemCassette object for this command.
ETAPIVERSION	″3″ (Indicates WebSphere Commerce Payments Version 2.1.x, 2.2.x and 3.1.x).
OBJECTNAME	ASCII character string. Value specified by the cassette. In conjunction with CASSETTENAME and protocol data parameters, uniquely identifies the target SystemCassette object for the command. The maximum length is 1000 bytes.
OPERATION	ASCII character string ″DeleteSystemCassetteObject″.

Optional keyword for DeleteSystemCassetteObject command.

Optional Keyword	Value
DTDPATH	Path to the locally-stored DTD. The value of this parameter is used in the XML document to specify the location of the external DTD. If this parameter is not specified, the complete DTD is returned as an internal DTD. The length of the DTDPath must be from 1 to 254 bytes.

Deposit

The DEPOSIT command results in the association of a specified payment with a batch and the subsequent deposit of previously approved monies for that payment. The DEPOSIT command is legal when operating on deposits in Approved state.

If successful, the specified payment is moved into Deposited state.

Required keywords for Deposit command

Required Keywords	Value
AMOUNT	Must be a 32-bit integer in ASCII characters.
ETAPIVERSION	″3″ (Indicates WebSphere Commerce Payments Version 2.1.x, 2.2.x and 3.1.x).
MERCHANTNUMBER	Integer in ASCII characters. Must be from 1 to 999999999.
OPERATION	ASCII character string ″Deposit.″
ORDERNUMBER	Integer in ASCII characters. Must be from 1 to 999999999.
PAYMENTNUMBER	Integer in ASCII characters. Must be from 1 to 999999999.

Optional keywords for Deposit command

Optional Keywords	Value
BATCHNUMBER	Identifies the batch under which this payment will be processed. Must be from 1 to 999999999.

Optional keywords for Deposit command

DTDPATH	Path to the locally stored DTD. The value of this parameter is used in the XML document to specify the location of the external DTD. If this parameter is not specified, the complete DTD is returned as an internal DTD. The length of the DTDPath must be from 1 to 254 bytes.

DepositReversal

A DEPOSITREVERSAL command disassociates a payment from a batch. This command is legal for payments in Deposited state. If successful, the payment moves to Approved state or Void state, and the deposited amount is reset to ″0″.

Required keywords for DepositReversal command

Required Keywords	Value
ETAPIVERSION	″3″ (Indicates WebSphere Commerce Payments Version 2.1.x, 2.2.x and 3.1.x).
MERCHANTNUMBER	Integer in ASCII characters. Must be from 1 to 999999999.
OPERATION	ASCII character string ″DepositReversal.″
ORDERNUMBER	Integer in ASCII characters. Must be from 1 to 999999999.
PAYMENTNUMBER	Integer in ASCII characters. Must be from 1 to 999999999.

Optional keywords for DepositReversal command

Optional Keyword	Value
DTDPATH	Path to the locally stored DTD. The value of this parameter is used in the XML document to specify the location of the external DTD. If this parameter is not specified, the complete DTD is returned as an internal DTD. The length of the DTDPath must be from 1 to 254 bytes.

ModifyAccount

The MODIFYACCOUNT command is used to change the attributes of a specified Account object.

Required keywords for ModifyAccount command

Required Keywords	Value
ACCOUNTNUMBER	Integer in ASCII characters. Must be from 1 to 999999999. In conjunction with MERCHANTNUMBER and CASSETTENAME, uniquely identifies the target Account for this command.

Required keywords for ModifyAccount command

CASSETTENAME	ASCII character string from 1 to 64 bytes. In conjunction with MERCHANTNUMBER and ACCOUNTNUMBER, uniquely identifies the target Account for this command.
ETAPIVERSION	″3″ (Indicates WebSphere Commerce Payments Version 2.1.x, 2.2.x and 3.1.x).
MERCHANTNUMBER	Integer in ASCII characters. Must be from 1 to 999999999. In conjunction with CASSETTENAME and ACCOUNTNUMBER, uniquely identifies the target Account for this command.
OPERATION	ASCII character string ″ModifyAccount″.

Optional keywords for ModifyAccount command

Optional Keywords	**Value**
ACCOUNTTITLE	UTF-8 string that is either null or from 1 to 254 bytes. If present, the value passed in will replace the AccountTitle specified Account object.
DTDPATH	Path to the locally-stored DTD. The value of this parameter is used in the XML document to specify the location of the external DTD. If this parameter is not specified, the complete DTD is returned as an internal DTD. The length of the DTDPath must be from 1 to 254 bytes.
ENABLED	ASCII character string ″0″ or ″1″, where ″0″ and ″1″ denote false and true, respectively. If present, the value passed in will replace the Enabled field of the Account object. Indicates whether the Account object should be active.
FINANCIALINSTITUTION	UTF-8 string that is either null or from 1 to 255 bytes. If present, the value passed in will replace the FinancialInstitution specified Account object.
APAPPROVEFLAG	Approve flag for AcceptPayment. ASCII character string ″0″, ″1″, or ″2″. Default is ″0″ ″0″ indicates transaction should not be approved. ″1″ indicates transaction should be approved automatically. ″2″ indicates transaction should be approved asynchronously.
RPAPPROVEFLAG	Approve flag for ReceivePayment. ASCII character string ″0″, ″1″, or ″2″. Default is ″0″ ″0″ indicates transaction should not be approved. ″1″ indicates transaction should be approved automatically. ″2″ indicates transaction should be approved asynchronously.

Optional keywords for ModifyAccount command

APDEPOSITFLAG	ASCII character string ″0″ or ″1″, where ″0″ and ″1″ denote false and true, respectively. Only specified if APAPPROVEFLAG is defined and not set to 0. Otherwise PRC_INVALID_PARAMETER_ COMBINATION_, RC_AP_DEPOSITFLAG will be returned.
RPDEPOSITFLAG	ASCII character string ″0″ or ″1″, where ″0″ and ″1″ denote false and true, respectively. Only specified if RPAPPROVEFLAG is defined and not set to 0. Otherwise PRC_INVALID_PARAMETER_ COMBINATION_, RC_RP_DEPOSITFLAG will be returned.
APPROVALEXPIRATION	Integer value that indicates the number of days after a payment has been approved that the approval expires. This field supports configurable approval expiration where this setting controls whether a payment approval associated with the account will expire after the elapsed time. A value of 0 implies no expiration. When a payment approval expires, it will be placed in the ApprovalExpired state. **Note:** A cassette is allowed to cause payment approvals to expire independently of this setting, but this parameter allows the framework to detect payment approval expiration on behalf of the cassette. For a description of the ApprovalExpired state see: “Payment states” on page 116

Note: APAPPROVEFLAG AND RPAPPROVEFLAG values are superceded by the API Approve flag when the API approve flag contains a non-zero, non-null value.

ModifyCassette

The MODIFYCASSETTE command is used to modify the properties of the specified cassette object.

Required keywords for ModifyCassette command

Required Keywords	Value
CASSETTENAME	ASCII character string from 1 to 64 bytes. Identifies the target cassette object for this command.
ETAPIVERSION	″3″ (Indicates WebSphere Commerce Payments Version 2.1.x, 2.2.x and 3.1.x).
OPERATION	ASCII character string ″ModifyCassette.″

Optional keywords for ModifyCassette command

Optional Keywords	Value
DTDPATH	Path to the locally-stored DTD. The value of this parameter is used in the XML document to specify the location of the external DTD. If this parameter is not specified, the complete DTD is returned as an internal DTD. The length of the DTDPath must be from 1 to 254 bytes.

Optional keywords for ModifyCassette command

ENABLED	ASCII character string ″0″ or ″1,″ where ″0″ and ″1″ denote false and true, respectively. If present, the value passed in will replace the Enabled field of the Cassette object. Indicates whether the Cassette object should be active.
TRACESETTING	32-bit bitmask. If present, the value passed in will replace the TraceSetting of the specified cassette object. See “Trace settings” for information on determining this value. Add all bits and use that number for TRACESETTING.

Trace settings

The following table provides a listing of various traces and their descriptions to aid you in providing the correct value for the TRACESETTING keyword.

Trace Type	Description	Bit of BITMASK
Connection Established	A new socket connection has been established	0x00000001
Connection Dropped/Lost	An existing socket connection has been closed	0x00000002
TCP Read	Data has been read from a TCP socket	0x00000004
TCP Read with data	Data has been read from a TCP socket (and includes trace)	0x00000008
TCP Write	Data has been written to a TCP socket	0x00000010
TCP Write with data	Data has been written to a TCP socket (and includes trace)	0x00000020
Function Entry	The specified function has just been entered	0x00000040
Function Exit	The specified function is about to be exited	0x00000080
API Command used	The specified API command was called	0x00000100
Database Read	Data has been read from the database	0x00000200
Database Write	Data has been written to the database	0x00000400
Database Commit	Changes have been committed to the database	0x00000800
Debug	A generic debug message	0x00001000
Object State change (order, batch payment, credit)	Used to output the state of an object	0x00002000
Error Occurred	An error has occurred	0x00004000
Start Work Item	The specified work item has been started	0x00008000
Trace Information Message	Used by the tracing code to output some informational messages (when tracing was started/stopped etc.)	0x00010000
System Info	General system information	0x00020000
Cassette Message 1	Reserved trace type for cassette-specific messages	0x80000000
Cassette Message 2	Reserved trace type for cassette-specific messages	0x40000000

Cassette Message 3	Reserved trace type for cassette-specific messages	0x20000000
Cassette Message 4	Reserved trace type for cassette-specific messages	0x10000000

Multiple types of traces can run at the same time; any combination of traces is legal. To determine the correct tracesetting value when running multiple traces, simply add the bits together. For example, if you are using all types of database tracing (e.g. DB_READ, DB_WRITE, and DB_COMMIT) you would need to add:

```
0x200 + 0x400 + 0x800 = 0xE00(3584)
```

In this example, 3584 would be the value used for TRACESETTING

If you wanted to use all of the cassette traces, you would need to add:

```
0x80000000 + 0x40000000 + 0x20000000 + 0x10000000 = 0xF0000000 (-268435456)
```

In this example -268435456 would be the value used for TRACESETTING.

TRACESETTING is a 32–bit signed integer. To summarize, the following procedure will aid you in determining the correct value for TRACESETTING:

1. Add up the bits to get X.
2. If 0≤ X ≤ 2147483647; use X for TRACESETTING
3. If X > 2147483647; X - 4294967296 is the value for TRACESETTING

ModifyMerchant

The MODIFYMERCHANT command modifies the properties of the specified Merchant object.

Required keywords for ModifyMerchant command

Required Keywords	Value
ETAPIVERSION	″3″ (Indicates WebSphere Commerce Payments Version 2.1.x, 2.2.x and 3.1.x).
MERCHANTNUMBER	Integer in ASCII characters. Must be from 1 to 999999999. Identifies the target Merchant object for the command.
OPERATION	ASCII character string ″ModifyMerchant.″

Optional keywords for ModifyMerchant command

Required Keywords	Value
DTDPATH	Path to the locally-stored DTD. The value of this parameter is used in the XML document to specify the location of the external DTD. If this parameter is not specified, the complete DTD is returned as an internal DTD. The length of the DTDPath must be from 1 to 254 bytes.
ENABLED	ASCII character string ″0″ or ″1,″ where ″0″ and ″1″ denotes false and true respectively. If present, the value passed in will replace the Enabled field of the Merchant object. Indicates whether the Merchant object should be active.

Optional keywords for ModifyMerchant command

MERCHANTTITLE	UTF-8 string that is either null or from 1 to 128 bytes present; the value passed in will replace the MerchantTitle specified Merchant object.

ModifyMerchantCassetteObject

The MODIFYMERCHANTCASSETTEOBJECT command modifies the properties of the Cassette object with type specified by the object name.

Refer to the appropriate cassette supplement for details on how this command is used.

Required keywords for ModifyMerchantCassetteObject command

Required Keywords	Value
CASSETTENAME	ASCII character string from 1 to 64 bytes. Specifies an identifier for the MerchantCassette object.
ETAPIVERSION	″3″ (Indicates WebSphere Commerce Payments Version 2.1.x, 2.2.x and 3.1.x).
MERCHANTNUMBER	Integer in ASCII characters. Must be from 1 to 999999999. In conjunction with CASSETTENAME, OBJECTNAME and protocol data parameters, it uniquely identifies the target MerchantCassette object for the command.
OBJECTNAME	ASCII character string. Value specified by the cassette. In conjunction with CASSETTENAME, MERCHANTNUMBER, and protocol data parameters, uniquely identifies the target MerchantCassette for this command. The maximum length is 1000 bytes.
OPERATION	ASCII character string ″ModifyMerchantCassetteObject.″

Optional keywords for ModifyMerchantCassetteObject command.

Optional Keywords	Value
DTDPATH	Path to the locally-stored DTD. The value of this parameter is used in the XML document to specify the location of the external DTD. If this parameter is not specified, the complete DTD is returned as an internal DTD. The length of the DTDPath must be from 1 to 254 bytes.
ENABLED	ASCII character string ″0″ or ″1,″ where ″0″ and ″1″ denote false and true, respectively. If present, the value passed in will replace the Enabled field of the MerchantCassette object. Indicates whether the MerchantCassette object should be enabled.

ModifyMerEventListener

The MODIFYMEREVENTLISTENER command modifies the specified MerEventListener object.

Required keywords for ModifyMerEventListener command

Required Keywords	Value
ENABLED	Can be set to 1(true) or 0 (false).
ETAPIVERSION	″3″ (Indicates WebSphere Commerce Payments Version 2.1.x, 2.2.x and 3.1.x).
EVENTTYPE	Integer in ASCII characters that identifies the event type. Events have the following values: 1: State change event 2: Cassette specific event
LISTENERURL	ASCII character string that identifies where the events show (for example, http://www.merchant.com/webapp/PaymentManager/eventReceiver888). If the port number is not specified, the default port number, 80, is used. No parameter limitations.
MERCHANTNUMBER	Integer in ASCII characters. Must be from 1 to 999999999.
OPERATION	ASCII character string ″ModifyMerEventListener.″

Optional keywords for ModifyMerEventListener command

Optional Keywords	Value
DTDPATH	Path to the locally-stored DTD. The value of this parameter is used in the XML document to specify the location of the external DTD. If this parameter is not specified, the complete DTD is returned as an internal DTD. The length of the DTDPath must be from 1 to 254 bytes.

Required keyword is EventType is set to 2.

Required Keywords	Value
CASSETTENAME	ASCII character string up to 64 bytes that identifies the cassette name. Required for modifying cassette events. No parameter limitations. Must match an existing cassette.

ModifyPayServer

The MODIFYPAYSERVER command modifies the global properties of the Payment Server object.

Required keywords for ModifyPayServer command

Required Keywords	Value
ETAPIVERSION	″3″ (Indicates WebSphere Commerce Payments Version 2.1.x, 2.2.x and 3.1.x).
OPERATION	ASCII character string ″ModifyPayServer.″

Optional keywords for ModifyPayServer command

Optional Keywords	Value

Optional keywords for ModifyPayServer command

DTDPATH	Path to the locally-stored DTD. The value of this parameter is used in the XML document to specify the location of the external DTD. If this parameter is not specified, the complete DTD is returned as an internal DTD. The length of the DTDPath must be from 1 to 254 bytes.
ENABLED	ASCII character string ″0″ or ″1″, where ″0″ and ″1″ denote false and true, respectively. If present, the value passed in will replace the Enabled field of the PayServer object. Indicates whether the PaymentServer object should be active.
ETILLHOSTNAME	ASCII character string, either null or from 1 to 254 characters present, the value passed in will replace the ETillHostname field in the PaymentServer object. A nonnull value indicates the DNS hostname that should be sent when sending messages to the WebSphere Commerce Payments. A null value indicates that DNS lookup should be used to determine the value.
LOGPATH	ASCII character string, either null or from 1 to 254 characters present, the value passed in will replace the LogPath of the PaymentServer object.
TRACEFILESIZE	ASCII character string representing a (64-byte) long, when it must be either null or positive. If present, the value passed will replace the TraceFileSize of the PaymentServer object. The minimum value is 4K (that is, 4096).
TRACESETTING	32-bit bitmask. If present, the value passed in will replace the TraceSetting of the PaymentServerObject. See “Trace settings” on page 83 for information on determining this value. Add all bits and use that number for TRACESETTING.

ModifyPaySystem

The MODIFYPAYSYSTEM command modifies the specified Payment System object.

Required keywords for ModifyPaySystem command

Required Keywords	**Value**
CASSETTENAME	ASCII character string from 1 to 64 bytes. In conjunction with MERCHANTNUMBER, uniquely identifies the target Payment System object command.
ETAPIVERSION	″3″ (Indicates WebSphere Commerce Payments Version 2.1.x, 2.2.x and 3.1.x).
MERCHANTNUMBER	Integer in ASCII characters. Must be from 1 to 999999999. In conjunction with CASSETTENAME, uniquely identifies the target PaymentSystem object command.
OPERATION	ASCII character string ″ModifyPaySystem″

Optional keywords for ModifyPaySystem command

Optional Keywords	**Value**

Optional keywords for ModifyPaySystem command

DTDPATH	Path to the locally-stored DTD. The value of this parameter is used in the XML document to specify the location of the external DTD. If this parameter is not specified, the complete DTD is returned as an internal DTD. The length of the DTDPath must be from 1 to 254 bytes.
ENABLED	ASCII character string ″0″ or ″1,″ where ″0″ and ″1″ denotes false and true, respectively. If present, the value passed in will replace the Enabled field of the ModifyPaySystem object. Indicates whether the ModifyPaySystem object should be active.

ModifySNMEventListener

The MODIFYSNMEVENTLISTENER command modifies the System Network Management Event Listener object.

Required keywords for ModifySNMEventListener command

Required Keywords	Value
ENABLED	Can be set to 1 (true) or 0 (false)
ETAPIVERSION	″3″ (Indicates WebSphere Commerce Payments Version 2.1.x, 2.2.x and 3.1.x).
EVENTTYPE	3: Identifies the SNM event type. Other values reserved for future use.
LISTENERURL	ASCII character string that identifies where the events show (for example, http://www.merchant.com/webapp/PaymentManager/eventReceiver888). If the port number is not specified, the default port number, 80, is used. A valid URL from 1 to 256 characters.
OPERATION	ASCII character string ″ModifySNMEventListener″

Optional keywords for ModifySNMEventListener command

Optional Keywords	Value
DTDPATH	Path to the locally-stored DTD. The value of this parameter is used in the XML document to specify the location of the external DTD. If this parameter is not specified, the complete DTD is returned as an internal DTD. The length of the DTDPath must be from 1 to 254 bytes.

ModifySystemCassetteObject

The MODIFYSYSTEMCASSETTEOBJECT command modifies the properties of the Cassette object with the type specified by object name.

Refer to the appropriate cassette supplement for details on how this command is used.

Required keywords for ModifySystemCassetteObject command

Required Keywords	Value

Required keywords for ModifySystemCassetteObject command

CASSETTENAME	ASCII character string from 1 to 64 bytes. In conjunction with OBJECTNAME and protocol data parameters, uniquely identifies the target SystemCassette object for this command.
ETAPIVERSION	″3″ (Indicates WebSphere Commerce Payments Version 2.1.x, 2.2.x and 3.1.x).
OBJECTNAME	ASCII character string. Value specified by the cassette. In conjunction with CASSETTENAME and protocol data parameters, uniquely the target SystemCassette object for this command. The maximum length is 1000 bytes.
OPERATION	ASCII character string ″ModifySystemCassetteObject.″

Optional keywords for ModifySystemCassetteObject command.

Optional Keywords	Value
DTDPATH	Path to the locally-stored DTD. The value of this parameter is used in the XML document to specify the location of the external DTD. If this parameter is not specified, the complete DTD is returned as an internal DTD. The length of the DTDPath must be from 1 to 254 bytes.
ENABLED	ASCII character string ″0″ or ″1,″ where ″0″ and ″1″ denotes false and true, respectively. If present, the value passed in will replace the Enabled field of the SystemCassette object. Indicates whether the SystemCassette object should be active.

ModifyUserStatus

This command changes the status of the user who has the access rights to the WebSphere Commerce Payments. Access control for this function is limited to the Payments administrators and the Merchant administrator. The Merchant administrator can only ″modify user status″ of the user in his merchant.

Required keywords for ModifyUserStatus command

Required Keywords	Value
ENABLED	Can be set to 1 (true) or 0 (false)
ETAPIVERSION	″3″ (Indicates WebSphere Commerce Payments Version 2.1.x, 2.2.x and 3.1.x).
MERCHANTNUMBER	String form of numeric merchant number. This keyword is required if any of the request is issued by a Merchant administrator.
OPERATION	ASCII character string ″ModifyUserStatus.″
USER	Byte array containing userid characters. ASCII character string from 1 to 80 characters.
ROLE	The value assigned to each WebSphere Commerce Payments role. For designated values, see Table 5 on page 103

QueryAccounts

The QUERYACCOUNTS command returns a collection of Account objects in XML format.

Required keywords and operational parameters for QueryAccounts command

Required Keyword	Multiple Allowed?	Value
ETAPIVERSION	N	″3″ (Indicates WebSphere Commerce Payments Version 2.1.x, 2.2.x and 3.1.x).
OPERATION	N	ASCII character string ″QueryAccounts.″

Optional operational parameter for QueryAccounts command

Optional Operational Parameter	Value
DTDPATH	Path to the locally-stored DTD. The value of this parameter will be used in the XML document to specify the location of the external DTD. If this parameter is not specified, the complete DTD will be returned as an internal DTD. The length of the DTDPath must be from 1 to 254 bytes.

Search modifiers for QueryAccounts command

Optional Keywords	Multiple Allowed?	Value
ACCOUNTNUMBER	Y	The account number. Integer in ASCII characters. Must be from 1 to 999999999.
CASSETTENAME	Y	The name of the cassette. ASCII character string with a maximum length of 64 bytes.
MERCHANTNUMBER	Y	The merchant number. Integer in ASCII characters. Must be from 1 to 999999999.

QueryBatches

The QUERYBATCHES command returns a collection of WebSphere Commerce Payments batch objects or batchkeys.

Required keywords and operational parameters for QueryBatches command

Required Keywords	Value
ETAPIVERSION	″3″ (Indicates WebSphere Commerce Payments Version 2.1.x, 2.2.x and 3.1.x).
OPERATION	ASCII character string ″QueryBatches.″

Optional operational parameters for QueryBatches command.

Optional Operational Parameter	Value
DTDPATH	ASCII character string. Path to the locally-stored DTD. The value of this parameter will be used in the XML document to specify the location of the external DTD. If this parameter is not specified, the complete DTD will be returned as an internal DTD. The length of the DTDPath must be from 1 to 254 bytes.

Optional operational parameters for QueryBatches command.

KEYSONLY	1: Instead of returning the actual objects, only a list of unique batch identifiers (in the form ″orderNumber:batchNumber″) should be returned. 0: The complete objects will be returned.
RETURNATMOST	Specifies the maximum number of objects or unique credit identifiers to return for this call. This enables the application to control the amount of data returned by a given query call. A 32-bit positive integer in ASCII characters.
WITHCREDITS	1: All related PSCredit objects should be located and kept with the batch objects. 0: Credits will not be returned.
WITHPAYMENTS	1: All related PSPayment objects should be located and kept with the batch objects. 0: Payments will not be returned.

Search modifiers for QueryBatches command.

Optional Search Modifiers	**Multiple Allowed?**	**Value**
ACCOUNTNUMBER	Y	Merchant's account with its financial institution. Integer in ASCII characters. Must be from 1 to 999999999.
BALANCESTATUS	Y	An ASCII character string containing one of the following values: ″batch_not_yet_balanced″ ″batch_balanced″ ″batch_out_of_balance″
BATCHNUMBER	Y	Integer in ASCII characters. Must be from 1 to 999999999.
CLOSEALLOWED	N	1: Only batches which the merchant is allowed to close should be returned. 0: Only batches that will be closed by the financial institution should be returned. If this parameter is not specified, or if any other value is specified, then both types of batches will be returned.
CLOSEBEGINTIME	N	A date and time to be used as the lower limit of the close time of the batch. To be included in the query result. This value is specified in ASCII decimal digits as the number of milliseconds since midnight (00:00:00:000 on a 24 hour clock), 01 January 1970.

Search modifiers for QueryBatches command.

CLOSEENDTIME	N	A date and time to be used as the upper limit of the close time of the batch. To be included in the query result. This value is specified in ASCII decimal digits as the number of milliseconds since midnight (00:00:00:000 on a 24 hour clock), 01 January 1970.
MERCHANTNUMBER	Y	Integer in ASCII characters. Must be from 1 to 999999999.
MODIFYBEGINTIME	N	A date and time to be used as the lower limit of the modify time of the batch. To be included in the query result. This value is specified in ASCII decimal digits as the number of milliseconds since midnight (00:00:00:000 on a 24 hour clock), 01 January 1970.
MODIFYENDTIME	N	A date and time to be used as the upper limit of the modify time of the batch. To be included in the query result. This value is specified in ASCII decimal digits as the number of milliseconds since midnight (00:00:00:000 on a 24 hour clock), 01 January 1970.
OPENBEGINTIME	N	A date and time to be used as the lower limit of the open time of the batch. To be included in the query result. This value is specified in ASCII decimal digits as the number of milliseconds since midnight (00:00:00:000 on a 24 hour clock), 01 January 1970.
OPENENDTIME	N	A date and time to be used as the upper limit of the open time of the batch. To be included in the query result. This value is specified in ASCII decimal digits as the number of milliseconds since midnight (00:00:00:000 on a 24 hour clock), 01 January 1970.
PAYMENTTYPE	Y	ASCII character string. Value has a maximum length of 10 bytes.
STATE	Y	An ASCII character string containing one of the following values: • ″batch_opening″ • ″batch_open″ • ″batch_closing″ • ″batch_closed″

QueryCassette

A QUERYCASSETTE command returns a collection of Cassette objects in XML format.

Required keywords and operational parameters for QueryCassettes command

Required Keyword	Value

Required keywords and operational parameters for QueryCassettes command

ETAPIVERSION	″3″ (Indicates WebSphere Commerce Payments Version 2.1.x, 2.2.x and 3.1.x).
OPERATION	ASCII character string ″QueryCassettes.″

Optional operational parameter for QueryCassettes command

Optional Operational Parameter	Value
DTDPATH	Path to the locally-stored DTD. The value of this parameter used in the XML document to specify the location of the external DTD. If this parameter is not specified, the complete DTD is returned as an internal DTD. The length of the DTDPath must be from 1 to 254 bytes.

Search modifiers for QueryCassettes command

Optional Search Modifiers	Multiple Allowed?	Value
CASSETTENAME	Y	The name of the cassette. ASCII character string with a maximum length of 64 bytes.

QueryCredits

The QUERYCREDITS command returns a collection of WebSphere Commerce Payments Credit objects or unique payment identifiers (in the form: ″orderNumber: creditNumber″).

Required keywords and operational parameters for QueryCredits command

Required Keyword	Value
ETAPIVERSION	″3″ (Indicates WebSphere Commerce Payments Version 2.1.x, 2.2.x and 3.1.x).
OPERATION	ASCII character string ″QueryCredits″.

Optional operational parameters for QueryCredits command.

Optional Operational Parameters	Value
DTDPATH	ASCII character string. Path to the locally-stored DTD. The value of this parameter will be used in the XML document to specify the location of the external DTD. If this parameter is not specified, the complete DTD will be returned as an internal DTD. The length of the DTDPath must be from 1 to 254 bytes.
KEYSONLY	1: Instead of returning the actual objects, only a list of unique credit identifiers (in the form ″merchantNumber:orderNumber:creditNumber″) should be returned. 0: The complete objects will be returned.
RETURNATMOST	Specifies the maximum number of objects or unique credit identifiers to return for this call. This enables the application to control the amount of data returned by a given query call. Integer in ASCII characters. 32-bit positive integer.

Optional operational parameters for QueryCredits command.

WITHORDERS	1: PSORDER object should be located and returned with the Credit objects. 0: Only Credit objects will be returned.

Search modifiers for QueryCredits command

Optional Search Modifiers	**Multiple Allowed?**	**Value**
BATCHNUMBER	Y	Integer in ASCII characters. Must be from 1 to 999999999.
BRAND	Y	Brand of customer's payment instrument. ASCII character string.
CREATEBEGINTIME	N	A date and time to be used as the lower limit of the create time of the credit. To be included in the query result. This value is specified in ASCII decimal digits as the number of milliseconds since midnight (00:00:00:000 on a 24 hour clock), 01 January 1970.
CREATEENDTIME	N	A date and time to be used as the upper limit of the create time of the credit. To be included in the query result. This value is specified in ASCII decimal digits as the number of milliseconds since midnight (00:00:00:000 on a 24 hour clock), 01 January 1970.
CREDITNUMBER	Y	Integer in ASCII characters. Must be from 1 to 999999999.
CURRENCY	N	The ISO 4217 currency code for amount values. Integer in ASCII characters. Must be exactly 3 characters long and should include leading zeroes if necessary.
MAXAMOUNT	N	Maximum credit amount. A Currency value must also be specified. A 32-bit positive integer in ASCII characters.
MERCHANTNUMBER	Y	Integer in ASCII characters. Must be from 1 to 999999999.
MINAMOUNT	N	Minimum credit amount. A Currency value must also be specified. A 32-bit positive integer in ASCII characters.
MODIFYBEGINTIME	N	A date and time to be used as the lower limit of the modify time of the credit. To be included in the query result. This value is specified in ASCII decimal digits as the number of milliseconds since midnight (00:00:00:000 on a 24 hour clock), 01 January 1970.

Search modifiers for QueryCredits command

MODIFYENDTIME	N	A date and time to be used as the upper limit of the modify time of the credit. To be included in the query result. This value is specified in ASCII decimal digits as the number of milliseconds since midnight (00:00:00:000 on a 24 hour clock), 01 January 1970.
ORDERNUMBER	Y	Integer in ASCII characters. Must be from 1 to 9999999999.
PAYMENTTYPE	Y	ASCII character string. Value has a maximum length of 10 characters.
REFERENCENUMBER	Y	Merchant-assigned reference number for this credit. ASCII character string.
STATE	Y	An ASCII character string containing one of the following values: • ″credit_reset″ • ″credit_refunded″ • ″credit_closed″ • ″credit_declined″ • ″credit_void″ • ″credit_pending″

QueryEventListeners

The QUERYEVENTLISTENERS command returns a collection of Event Listener objects.

Required keyword for QueryEventListeners command

Required Keyword	Multiple Allowed?	Value
ETAPIVERSION	N	″3″ (Indicates WebSphere Commerce Payments Version 2.1.x, 2.2.x and 3.1.x).
OPERATION	N	ASCII character string ″QueryEventListeners.″

Optional operational parameters for QueryEventListeners command

Optional Keywords	Value
DTDPATH	Path to the locally-stored DTD. The value of this parameter is used in the XML document to specify the location of the external DTD. If this parameter is not specified, the complete DTD is returned as an internal DTD. The length of the DTDPath must be from 1 to 254 bytes.

Search modifiers for QueryEventListeners command

Optional Search Modifiers	Multiple Allowed?	Value
CASSETTENAME	Y	ASCII character string, 1 to 64 bytes..

Search modifiers for QueryEventListeners command

EVENTTYPE	Y	Integer in ASCII characters. Value must be from 1 to 3: • 1 = state change event • 2 = cassette event • 3 = network management event
LISTENERURL	Y	ASCII character string that identifies where the events show (for example, http://www.merchant.com/webapp/PaymentManager /eventReceiver888). If the port number is not specified, the default port number, 80, is used. No parameter limitations.
MERCHANTNUMBER	Y	Integer in ASCII characters. Value must be from 1 to 999999999.

QueryMerchants

The QUERYMERCHANTS command returns a collection of Merchant objects.

Required keywords and operational parameters for QueryMerchants command

Required Keywords	Multiple Allowed?	Value
ETAPIVERSION	N	″3″ (Indicates WebSphere Commerce Payments Version 2.1.x, 2.2.x and 3.1.x).
OPERATION	N	ASCII character string ″QueryMerchants.″

Optional operational parameter for QueryMerchants command.

Optional Operational Parameter	Value
DTDPATH	ASCII character string. Path to the locally-stored DTD. The value of this parameter will be used in the XML document to specify the location of the external DTD. If this parameter is not specified, the complete DTD will be returned as an internal DTD. The length of the DTDPath must be from 1 to 254 bytes.

Search modifier for QueryMerchants command

Optional Search Modifier	Multiple Allowed?	Value
MERCHANTNUMBER	Y	The merchant number. If no merchant number is specified, PSMerchant elements will be returned for all merchants defined to the WebSphere Commerce Payments. Integer in ASCII characters. Must be from 1 to 999999999.

QueryOrders

The QUERYORDERS command returns a collection of PSOrder objects or order numbers.

Required keywords and operational parameters for QueryOrders command

Required Keywords	Value
ETAPIVERSION	″3″ (Indicates WebSphere Commerce Payments Version 2.1.x, 2.2.x and 3.1.x).
OPERATION	ASCII character string ″QueryOrders.″

Optional operational parameters for QueryOrders command.

Optional Operational Parameters	Value
DTDPATH	ASCII character string. Path to the locally-stored DTD. The value of this parameter will be used in the XML document to specify the location of the external DTD. If this parameter is not specified, the complete DTD will be returned as an internal DTD. The length of the DTDPath must be from 1 to 254 bytes.
KEYSONLY	1: Instead of returning the actual objects, only a list of order numbers and merchant numbers should be returned. 0: Complete objects will be returned.
RETURNATMOST	Specifies the maximum number of objects or order numbers to return for this call. Enables the application to control the amount of data returned by a given query call. A 32-bit positive integer in ASCII characters.
WITHCREDITS	1: All related PSCredit objects should be located and kept with the Order objects. 0: Credits will not be returned.
WITHPAYMENTS	1: All related PSPayment objects should be located and kept with the Order objects. 0: Payments will not be returned.

Search modifiers for QueryOrders command

Optional Search Modifiers	Multiple Allowed?	Value
ACCOUNTNUMBER	Y	Merchant's account with its financial institution. Integer in ASCII characters. Must be from 1 to 999999999.
APPROVESALLOWED	N	Supported values are: 1: Approve command is allowed for this order 0: Approve command is not allowed for this order
BRAND	Y	Brand of customer's payment instrument. ASCII character string

Search modifiers for QueryOrders command

CREATEBEGINTIME	N	A date and time to be used as the lower limit of the create time of the order. To be included in the query result. This value is specified in ASCII decimal digits as the number of milliseconds since midnight (00:00:00:000 on a 24 hour clock), 01 January 1970.
CREATEENDTIME	N	A date and time to be used as the upper limit of the create time of the order. To be included in the query result. This value is specified in ASCII decimal digits as the number of milliseconds since midnight (00:00:00:000 on a 24 hour clock), 01 January 1970.
CURRENCY	N	The ISO 4217 currency code for amount values. Integer in ASCII characters. Must be exactly 3 characters long and should include leading zeroes if necessary.
MAXAMOUNT	N	Maximum order amount. A Currency value must also be specified. A 32-bit positive integer in ASCII characters.
MAXUNAPPROVEDAMOUNT	N	Maximum order amount that has yet to be approved. A Currency value must also be specified. A 32-bit positive integer in ASCII characters.
MERCHANTNUMBER	Y	Merchant number. Integer must be in ASCII characters. Value must be from 1 to 999999999.
MINAMOUNT	N	Minimum order amount. A Currency value must also be specified. A 32-bit positive integer in ASCII characters.
MINUNAPPROVEDAMOUNT	N	Minimum order amount that has yet to be approved. A Currency value must also be specified. A 32-bit positive integer in ASCII characters.
MODIFYBEGINTIME	N	A date and time to be used as the lower limit of the modify time of the order. To be included in the query result. This value is specified in ASCII decimal digits as the number of milliseconds since midnight (00:00:00:000 on a 24 hour clock), 01 January 1970.
MODIFYENDTIME	N	A date and time to be used as the upper limit of the modify time of the order. To be included in the query result. This value is specified in ASCII decimal digits as the number of milliseconds since midnight (00:00:00:000 on a 24 hour clock), 01 January 1970.
ORDERDATA1	N	Auxiliary data supplied by the user, specified as an ASCII character string between 1 and 254 bytes in length.
ORDERNUMBER	Y	Integer in ASCII characters. Must be from 1 to 99999999.

Search modifiers for QueryOrders command

PAYMENTTYPE	Y	Payment system type. Integer in ASCII characters. Maximum length is 10 bytes
STATE	Y	An ASCII character string containing one of the following values: • ″order_requested″ • ″order_ordered″ • ″order_refundable″ • ″order_rejected″ • ″order_pending″
TRANSACTIONID	N	Transaction identifier supplied by the user, specified as an ASCII character string from 1 to 128 bytes in length.

QueryPayments

The QUERYPAYMENTS command returns a collection of WebSphere Commerce Payments Payment objects or unique payment identifiers (in the form ″orderNumber: paymentNumber″).

Required keywords and operational parameters for QueryPayments command

Required Keywords	**Value**
ETAPIVERSION	″3″ (Indicates WebSphere Commerce Payments Version 2.1.x, 2.2.x and 3.1.x).
OPERATION	ASCII character string ″QueryPayments.″

Search modifiers for QueryPayments command

Optional Search Modifiers	**Multiple Allowed?**	**Value**
BATCHNUMBER	Y	Integer in ASCII characters. Must be from 1 to 999999999.
BRAND	Y	Brand of customer's payment instrument. ASCII character string.
CREATEBEGINTIME	N	A date and time to be used as the lower limit of the create time of the payment. To be included in the query result. This value is specified in ASCII decimal digits as the number of milliseconds since midnight (00:00:00:000 on a 24 hour clock), 01 January 1970.
CREATEENDTIME	N	A date and time to be used as the upper limit of the create time of the payment. To be included in the query result. This value is specified in ASCII decimal digits as the number of milliseconds since midnight (00:00:00:000 on a 24 hour clock), 01 January 1970.
CURRENCY	N	The ISO 4217 currency code for amount values. Integer in ASCII characters. Must be exactly 3 characters long and should include leading zeroes if necessary.

Search modifiers for QueryPayments command

MAXAPPROVEAMOUNT	N	Maximum approved amount. A Currency value must also be specified. A 32-bit positive integer in ASCII characters.
MAXDEPOSITAMOUNT	N	Maximum deposit amount. A Currency value must also be specified. A 32-bit positive integer in ASCII characters.
MERCHANTNUMBER	Y	Integer in ASCII characters. Must be from 1 to 999999999.
MINAPPROVEAMOUNT	N	Minimum approved amount. A Currency value must also be specified. A 32-bit positive integer in ASCII characters.
MINDEPOSITAMOUNT	N	Minimum deposit amount. A Currency value must also be specified. A 32-bit positive integer in ASCII characters.
MODIFYBEGINTIME	N	A date and time to be used as the lower limit of the modify time of the payment. To be included in the query result. This value is specified in ASCII decimal digits as the number of milliseconds since midnight (00:00:00:000 on a 24 hour clock), 01 January 1970.
MODIFYENDTIME	N	A date and time to be used as the upper limit of the modify time of the payment. To be included in the query result. This value is specified in ASCII decimal digits as the number of milliseconds since midnight (00:00:00:000 on a 24 hour clock), 01 January 1970.
ORDERNUMBER	Y	Integer in ASCII characters. Must be from 1 to 999999999.
PAYMENTNUMBER	Y	Integer in ASCII characters. Must be from 1 to 999999999.
PAYMENTTYPE	Y	Integer in ASCII characters. Maximum length is 10 bytes.
REFERENCENUMBER	Y	Merchant-assigned reference number for this payment. ASCII character string.
STATE	Y	An ASCII character string containing one of the following values: • ″payment_reset″ • ″payment_approved″ • ″payment_deposited″ • ″payment_closed″ • ″payment_declined″ • ″payment_void″ • ″payment_pending″

Optional operational parameters for QueryPayments command.

Optional Operational Parameters	**Value**

Optional operational parameters for QueryPayments command.

DTDPATH	ASCII character string. Path to the locally-stored DTD. The value of this parameter will be used in the XML document to specify the location of the external DTD. If this parameter is not specified, the complete DTD will be returned as an internal DTD. The length of the DTDPath must be from 1 to 254 bytes.
KEYSONLY	1: Instead of returning the actual objects, only a list of unique payment identifiers (in the form ″merchantNumber: orderNumber: paymentNumber″) should be returned. 0: The complete objects will be returned.
RETURNATMOST	Specifies the maximum number of objects or unique payment identifiers to return for this call. This enables the application to control the amount of data returned by a given query call. A 32-bit positive integer in ASCII characters.
WITHORDERS	1: PSOrder object should be located and returned with the payment objects. 0: Order will not be returned.

QueryPaymentServer

The QUERYPAYMENTSERVER command returns the Payment Server object.

Required keywords and operational parameter for QueryPaymentServer command

Required Keyword	Multiple Allowed?	Value
ETAPIVERSION	N	″3″ (Indicates WebSphere Commerce Payments Version 2.1.x, 2.2.x and 3.1.x).
OPERATION	N	ASCII character string ″QueryPaymentServer.″

Optional operational parameter for QueryPaymentServer command

Optional Operational Parameter	Value
DTDPATH	Path to the locally-stored DTD. The value of this parameter used in the XML document to specify the location of the external DTD. If this parameter is not specified, the complete DTD is returned as an internal DTD. The length of the DTDPath must be from 1 to 254 bytes.

QueryPaySystems

The QUERYPAYSYSTEMS command returns a collection of Payment System objects.

Required keywords and operational parameters for QueryPaySystems command

Required Keyword	Multiple Allowed?	Value

Required keywords and operational parameters for QueryPaySystems command

ETAPIVERSION	N	″3″ (Indicates WebSphere Commerce Payments Version 2.1.x, 2.2.x and 3.1.x).
OPERATION	N	ASCII character string ″QueryPaySystems.″

Optional operational parameter for QueryPaySystems command

Optional Operational Parameter	Value
DTDPATH	Path to the locally-stored DTD. The value of this parameter will be used in the XML document to specify the location of the external DTD. If this parameter is not specified, the complete DTD will be returned as an internal DTD. The length of the DTDPath must be from 1 to 254 bytes.

Search modifiers for QueryPaySystems command

Optional Search Modifiers	Multiple Allowed?	Value
CASSETTENAME	Y	The cassette name. ASCII character string. Maximum length is 64 bytes.
MERCHANTNUMBER	Y	The merchant number. Integer in ASCII characters. Must be from 1 to 999999999.

QueryUsers

The QUERYUSERS command returns a collection of User objects.

Optional parameters

MerchantNumber
Performing QUERYUSERS on MerchantNumber returns all users associated with that merchant.

Filter The QUERYUSERS command enables administrators to query users by specifying a user *filter*. The filter is used by each realm to identify a subset of the whole user registry. The actual filter semantics must be defined by each realm. Both the PSDefaultRealm and the PSOS400Realm allow the filter to specify the character substrings of the username. For example, calling QUERYUSERS and passing a filter of `Smi` might result in a list of users including `Smith`, `Smitty` and `Jones-Smittinger`. Note that both the PSDefaultRealm and the PSOS400Realm will treat the user filter as case insensitive. The filter parameter specifies a *filter* to screen the users being returned. For more information, refer to “Valid combination of parameters” on page 103.

Note that when the Merchant administrator requires additional userids, they must be created and assigned by the Payments administrator.

The following table details the command syntax for the QUERYUSERS command:

Table 4. Optional keywords for QueryUsers command

Optional Keywords	Multiple Allowed?	Value

Table 4. Optional keywords for QueryUsers command (continued)

ETAPIVERSION	N	″3″ (Indicates WebSphere Commerce Payments Version 2.1.x, 2.2.x and 3.1.x)
MERCHANTNUMBER	Y	String form of numeric merchant number.
OPERATION	N	ASCII character string ″QueryUsers.″
ROLE	N	The value assigned to each WebSphere Commerce Payments role. For designated values, see Table 5 below.
USER	N	Maximum length is 80 bytes. This is the user name.
RETURNATMOST	N	Integer in ASCII characters. 32–bit positive integer. The maximum number of users to be returned is 10000.
FILTER	N	ASCII character string with a maximum length of 128 bytes.

Table 5. Role Values and Specifications

Value	Meaning	Merchant-specific Role?
0	Payments Administrator	N
1	Merchant Administrator	Y
2	Supervisor	Y
3	Clerk	Y

Valid combination of parameters

The following table illustrates all parameter combinations for the QUERYUSERS command. It also maps who can issue commands for the parameter combinations and what results will be returned.

Note that in most cases, WebSphere Commerce Payments does not check for duplicate parameters. If more than one instance of a parameter is specified, then the last instance will be used.

Table 6. Valid parameter combinations for QueryUsers

Parameter combinations	Valid?	Who can issue?	Return unauthorized users
No parameters specified	Yes	PMA	Yes
MERCHANTNUMBER	Yes	PMA/MA	No
ROLE	Yes	PMA	No
USER	Yes	All	Yes
MERCHANTNUMBER + ROLE	Yes	PMA/MA	No
MERCHANTNUMBER + USER	Yes	All	No
ROLE + USER	Yes	All	No
MERCHANTNUMBER + ROLE + USER	Yes	All	No
FILTER	Yes	PMA	Yes
FILTER + MERCHANTNUMBER	Yes	PMA/MA	No
FILTER + ROLE	Yes	PMA	No
FILTER + MERCHANTNUMBER + ROLE	Yes	PMA/MA	No
FILTER + USER	Yes, but filter will be ignored	All	Yes
FILTER + MERCHANTNUMBER + USER	Yes, but filter will be ignored	All	No
FILTER + ROLE + USER	Yes, but filter will be ignored	All	No
FILTER + MERCHANTNUMBER + USER + ROLE	Yes, but filter will be ignored	All	No

Parameter combinations

Some key points about QUERYUSERS parameter combinations:

- When the Username is specified, the filter will be ignored.
- To return the unauthorized users, you can use only one of the following methods:
  1. Use the filter without the Username
  2. Do not specify any parameters
  3. Query with Username only

Valid Though a parameter combination may be defined in the QUERYUSERS parameter table as being valid, certain queries may still be invalid. For example, even though a merchant administrator can issue a query with Role and Username parameters, the query will be allowed only when the username specified is the merchant administrator's username (that is, when the merchant administrator is querying himself). For more details on access control for the QUERYUSERS command, see "Access control details" on page 105.

Return unauthorized users

The *Return unauthorized users* column indicates whether the specified parameter combination can return users who are in the realm, but are not authorized to use the WebSphere Commerce Payments. This allows Payments administrators to query a single user and assign that user WebSphere Commerce Payments access. Note that all calls to QUERYUSERS can return users who *are* authorized.

Note that a realm may choose not to return all the matching users in the realm - especially if the filter is very unrestrictive. In these cases, the above methods will set the User objectCount to the total number of matching realm users. This, in turn, will indicate to the QUERYUSERS caller that the results are not complete and that a more restrictive search filter should be applied.

Access control details

. Whether a query is allowed is dependent on the role of the query issuer.

Payments administrator

The Payments administrator can issue a query with any combination of the parameters.

Merchant administrator

A merchant administrator can only query users who:

- are associated with a merchant number (or numbers) that is managed by the merchant administrator

In addition, the merchant administrator needs to adhere to the following requirements in his query request:

- At least one MerchantNumber needs to be specified, and all of the merchant numbers specified should belong to merchants associated with the merchant administrator. There is one exception where the merchant number is not required: the merchant administrator queries himself.
- If the Role parameter is specified, it should not contain the role of the Payments administrator.

Supervisors and clerks

For all other roles, the user can query himself. In this case, if the filter is specified, the filter will be ignored.

ReceivePayment

The RECEIVEPAYMENT command is used for order creation when there is electronic wallet participation. If successful, the order object is moved to Requested state. Subsequent wallet communication will complete the order and move it to Ordered state.

When creating an order, you may want to approve or deposit funds automatically. Once wallet communication is done and the order is in Ordered state, the APPROVEFLAG and DEPOSITFLAG keywords indicate that a Payment object should be automatically deposited and approved. Refer to the appropriate table below for additional keywords that are used if APPROVEFLAG or DEPOSITFLAG are specified.

Table 7. Required keywords for ReceivePayment command

Required Keywords	Value
AMOUNT	Must be 32-bit positive integer in ASCII characters.
CURRENCY	Integer in ASCII characters. See Appendix B, Currency Codes, for a list of ISO currency codes.
ETAPIVERSION	″3″ (Indicates WebSphere Commerce Payments Version 2.1.x, 2.2.x and 3.1.x).
MERCHANTNUMBER	Integer in ASCII characters. Must be from 1 to 999999999.
OPERATION	ASCII character string ″ReceivePayment.″

Table 7. Required keywords for ReceivePayment command (continued)

ORDERNUMBER	Integer in ASCII characters. Must be from 1 to 999999999.
PAYMENTTYPE	ASCII character string. Specifies the payment protocol being used; for example, OfflineCard.

Table 8. Optional keywords for ReceivePayment command

Optional Keywords	**Value**
AMOUNTEXP10	Integer in ASCII characters. Indicates the number of decimal places to shift. For more information on this keyword, refer to “Using the AmountExp10 keyword” on page 62.
APPROVEFLAG	Integer in ASCII characters. Indicates whether the approvals should be attempted automatically. Default is 0. Supported values are: 0 - Indicates transaction should not be approved. 1 - Indicates transaction should be approved automatically. 2 - Indicates transaction should be approved asynchronously.
DTDPATH	Path to the locally stored DTD. The value of this parameter is used in the XML document to specify the location of the external DTD. If this pbarameter is not specified, the complete DTD is returned as an internal DTD. The length of the DTDPath must be from 1 to 254 bytes.
ORDERDATA1	Auxiliary data supplied by the user, specified as an ASCII character string between 1 and 254 bytes in length.
ORDERDATA2	Auxiliary data supplied by the user, specified as a UTF-8 string from 1 to 254 bytes in length.
ORDERDATA3	Auxiliary data supplied by the user, specified as a UTF-8 string between 1 and 254 bytes in length.
ORDERDATA4	Auxiliary data supplied by the user, specified as a binary string between 1 and 254 bytes in length.
ORDERDATA5	Auxiliary data supplied by the user, specified as a binary string with an arbitrary length.
ORDERURL	URL containing order details.
TRANSACTIONID	Transaction identifier supplied by the user, specified as an ASCII character string between 1 and 128 bytes in length.

The following tables list the required and optional keywords for APPROVEFLAG=1 or 2.

Table 9. Required keywords if APPROVEFLAG is set to 1 or 2

Required Keywords	**Value**
PAYMENTAMOUNT	A 32–bit positive integer in ASCII characters.
PAYMENTNUMBER	Integer in ASCII characters. Must be from 1 to 999999999.

Table 10. Optional keywords if APPROVEFLAG is set to 1 or 2.

Optional Keywords	**Value**

Table 10. Optional keywords if APPROVEFLAG is set to 1 or 2. (continued)

DEPOSITFLAG	Boolean value in ASCII characters. Indicates whether the deposit should be attempted automatically. This flag is only valid if APPROVE=1 (order is automatically approved). Supported values are: 0 - Funds should not be automatically deposited 1 - Funds should be automatically deposited.

If DEPOSITFLAG=1, then the following keyword is optional:

Table 11. Optional keyword if DEPOSITFLAG is set to 1

Optional Keywords	Value
BATCHNUMBER	Identifies the batch under which this payment will be processed. Must be from 1 to 999999999.

Refund

A REFUND command is used to create a Credit object and is used when the merchant wants to return monies to the cardholder. The REFUND command is legal when the specified order is in Refundable state.

If successful, a Credit object will be created in Refunded or Closed state. If unsuccessful, a Credit object will be in Declined state.

Table 12. Required keywords for Refund command

Required Keywords	Value
AMOUNT	Must be a 32-bit positive integer in ASCII characters.
CREDITNUMBER	Integer in ASCII characters. Must be from 1 to 999999999. Indicates the number assigned to this credit.
ETAPIVERSION	″3″ (Indicates WebSphere Commerce Payments Version 2.1.x, 2.2.x and 3.1.x).
MERCHANTNUMBER	Integer in ASCII characters. Must be from 1 to 999999999.
OPERATION	ASCII character string ″Refund.″
ORDERNUMBER	Integer in ASCII characters. Must be from 1 to 999999999.

Table 13. Optional keywords for Refund command

Optional Keywords	Value
BATCHNUMBER	Optional for implicit batch. A numeric string of up to nine characters. Identifies the batch under which this payment will be processed. Must be from 1 to 999999999.
DTDPATH	Path to the locally stored DTD. The value of this parameter is used in the XML document to specify the location of the external DTD. If this parameter is not specified, the complete DTD is returned as an internal DTD. The length of the DTDPath must be from 1 to 254 bytes.

RefundReversal

A REFUNDREVERSAL command is used to void existing Credit objects. This command operates on Credit objects in Refunded state. A successful REFUNDREVERSAL call will result in the Credit object moving to Void State. If unsuccessful, the Credit object remains in Refunded state.

Table 14. Required keywords for RefundReversal command

Required Keywords	Value
CREDITNUMBER	Integer in ASCII characters. Must be from 1 to 999999999. Indicates the number assigned to this credit.
ETAPIVERSION	″3″ (Indicates WebSphere Commerce PaymentsVersion 2.1.x, 2.2.x and 3.1.x).
MERCHANTNUMBER	Integer in ASCII characters. Must be from 1 to 999999999.
OPERATION	ASCII character string ″RefundReversal.″
ORDERNUMBER	Integer in ASCII characters. Must be from 1 to 999999999.

Table 15. Optional keywords for RefundReversal command

Optional Keywords	Value
DTDPATH	Path to the locally stored DTD. The value of this parameter is used in the XML document to specify the location of the existing DTD. If this parameter is not specified, the complete DTD is returned as an internal DTD. The length of the DTDPath must be from 1 to 254 bytes.

SetUserAccessRights

The SETUSERACCESSRIGHTS command is used to set, change, or remove a user's access rights. However, this command will not create or remove users from the realm you are using to authenticate users. Before using the SetUserAccessRights command, make sure the user has been added to your realm (for example, PSOS400Realm or PSDefaultRealm).

adding user access
: If you want to add a user's access rights, first add that particular user to the realm and then issue the SetUserAccessRights command.

removing user access
: If you want to remove the user's access rights, issue the SetUserAccessRights command first to remove the user's access rights and then remove the user from the realm.

Table 16. Required keywords for SetUserAccessRights command

Required Keywords	Value
ETAPIVERSION	″3″ (Indicates WebSphere Commerce PaymentsVersion 2.1.x, 2.2.x and 3.1.x).
MERCHANTNUMBER	String form of numeric merchant number. This keyword is required if any of the roles specified is merchant specific. Merchant number must be from 1 to 999999999. For users other than the Payments administrator, multiple keyword-value pairs can be specified.
OPERATION	ASCII character string ″SetUserAccessRights.″

Table 16. Required keywords for SetUserAccessRights command (continued)

ROLE	String form of numeric value.
USER	ASCII character string with a maximum length of 40 bytes. (Note that a user may not update himself. That is to say, user ″admin″ may not call SETUSERACCESSRIGHTS with the user parameter set to ″admin″.)

To set or change a user's access rights, specify the role and the merchant number(s) on the command. To set or change a user's access rights such that the user has a role with *multiple* merchants, you must repeat the keyword-value pairs of the merchant number multiple times. The merchant number(s) must be specified if any role given is merchant-specific (See Table 5 on page 103) and must not be specified if the role given is non-merchant-specific.

Notes:

1. If the Role parameter is not specified, this command can be used to remove a user's access rights. In which case, the WebSphere Commerce Payments will ignore the merchant numbers (even though they are specified in the command).
2. A user may not update himself. That is to say, user ″admin″ may not call SETUSERACCESSRIGHTS with the user parameter set to ″admin″.

Access control rules for merchant administrators

Only the Payments administrator and the merchant administrator can assign or change a user's permission (or role). The Payments administrator can assign or change *any* user's access rights and can assign or change a user's role to whatever he wants that user's role to be, including the role of Payments administrator. Whereas the merchant administrator can assign or remove a user as a merchant administrator, supervisor, or clerk, he cannot assign or change a user's permissions to that of Payments administrator. Further, the merchant administrator can assign and change permissions only under the conditions outlined in "Assigning a user's access permissions" on page 14.

Chapter 8. WebSphere Commerce Payments data

This chapter focuses on WebSphere Commerce Payments Framework payment and administration objects and states. An object is a collection of data maintained by WebSphere Commerce Payments which represents a real-world entity. Each object is defined, and tables are provided to indicate field names, syntax and descriptions. The state of an object provides information on legal actions for that particular object. Query commands can be used to retrieve the current state of an object. Additional tables list the possible states of a particular object, along with a description of what that state means and which commands are legal for that state.

WebSphere Commerce Payments payment objects

WebSphere Commerce Payments defines the following Framework objects for all electronic payments, regardless of payment protocol:

- Order
- Payment
- Credit
- Batch

WebSphere Commerce Payments uses the terms *order*, *payment*, and *credit* to represent payment data for all electronic payment. An Order is an object that is created as a result of a data flow between a buyer and a merchant, while the buyer is placing an order for merchandise or services. Transactions flow between the merchant and the financial institution during the Order life cycle. These transactions can be broken into two broad categories: *payments* (monies transferred to the merchant from the consumer) and *credits* (monies returned to the buyer, such as when merchandise is defective). As processing on an Order continues, Payment and Credit objects are created, modified, and destroyed.

Another type of object used by the WebSphere Commerce Payments is a *batch* object. A batch represents multiple transactions processed as a group, such as the deposit of all payments at the end of a business day. Batch objects in the WebSphere Commerce Payments keep track of the collections of transactions. For instance, if a financial institution tells the merchant to close out the week's transactions, the merchant will close the current batch and open a new one. Batch objects for these two batches will reflect the new status of the batches.

Order, Payment, Credit, and Batch objects each have an associated *state*. The state of an object determines what actions are *permitted* for the object. The state of an object is determined by the action, or *command*, that was last performed on it.

Each WebSphere Commerce Payments Framework object is defined by its attributes, or fields. In the sections that follow, Object Tables display field names, field syntax, and field descriptions for each Framework object. In addition, Object State Tables display the states an object can assume and field descriptions for those states.

Order

An Order represents all the instructions and information needed from the buyer (payer) in order for the merchant (payee) to collect money. The merchant may collect that money all at once, or over a period of time, but never needs to go back to the buyer for additional information. The required information is all there in the

Order. The WebSphere Commerce Payments Order object describes the data included in the order. Each Order can have zero or more payments associated with it. The attributes for the Order object are:

Table 17. PSOrderObject Attributes

Field Name	Syntax	Description
merchantNumber	Numeric token, 1 to 9 digits long	A number that identifies the merchant that created the Order.
orderNumber	Numeric token, 1 to 9 digits long	A number assigned by the merchant that uniquely identifies the Order.
merchantOriginated	0 or 1 (Boolean)	Value is 1, *(true)* if the Order was created using AcceptPayment. Value is 0, *(false)* if the Order was created using ReceivePayment.
amount	Integer	Identifies the Order amount in the smallest denomination of the particular currency used to place the Order. When combined with AmountExp10, this field specifies the amount of the full Order in the specified currency.
amountExp10	Integer	Indicates the number of decimal places to shift the decimal point to reflect the currency. For example, if the amount is 2325, the currency code is for U.S. dollars, and AmountExp10 is -2, the transaction amount in U.S. dollars is $23.25.
currency	Integer	ISO code for currency. For example, 840 is the numeric code for a U.S. dollar, and 392 is the numeric code for a Japanese yen.
paymentType	Character string	Identifies the payment type, or protocol, used to place the Order (for example, SET or OfflineCard).
timeStampCreated	Date	The time that this Order entry was created. The number of milliseconds since midnight January 1, 1970 GMT.
timeStampModified	Date	The time that this Order entry was last modified. The number of milliseconds since midnight, January 1, 1970 GMT.
state	Character string	The state of the Order. • order_requested • order_ordered • order_refundable • order_rejected • order_pending • order_canceled • order_closed
approvesAllowed	0 or 1 (Boolean)	Flag indicating if approve commands are legal on this Order.
unapprovedAmount	Integer	Amount of the Order minus the approved amount of all Payments for that Order.
numberOfPayments	Integer	The number of payments associated with this Order.
numberOfCredits	Integer	The number of credits associated with this order.
brand	Character string	For credit card payment types: the payment card brand used to place this Order (for example, VISA or MasterCard).
orderURL	URL	A merchant-defined URL often used to point to information about the Order in the merchant's business system.
merchantAccount	Numeric token, 1 to 9 digits long	The number of the Account used to process this Order. Assigned prior to the Order entering Ordered state.

Table 17. PSOrderObject Attributes (continued)

Field Name	Syntax	Description
transactionId	Character string, 1 to 128 ASCII characters long	Customer's transaction identifier. This value will only be present if a non-null TRANSACTIONID value was specified on the AcceptPayment or ReceivePayment command.
orderData1	Character string, 1 to 254 ASCII characters long	This value will only be present if a non-null ORDERDATA1 value was specified on the AcceptPayment or ReceivePayment command.
orderData2	UTF-8 string, 1 to 254 bytes long	This value will only be present if a non-null ORDERDATA2 value was specified on the AcceptPayment or ReceivePayment command.
orderData3	UTF-8 string, 1 to 254 bytes long	This value will only be present if a non-null ORDERDATA3 value was specified on the AcceptPayment or ReceivePayment command.
orderData4	Binary string, 1 to 254 bytes long	This value will only be present if a non-null ORDERDATA4 value was specified on the AcceptPayment or ReceivePayment command.
orderData5	Binary string of an arbitrary length	This value will only be present if a non-null ORDERDATA5 value was specified on the AcceptPayment or ReceivePayment command.

Note: A *numeric token* is defined as a numeric string that is one to nine digits in length.

Order states

The state of an object determines what actions are *legal* for the object. The state of an object is determined by the action, or *command*, that was last performed on it (for example, a Payment that was approved, moves into Approved state).

Orders are in one of the following states:

State	Description
Requested	A preliminary state where the buyer has not yet provided all of the information necessary to complete the Order. Legal commands for this state: • CancelOrder

State	Description
Ordered	Indicates consumer/merchant server/WebSphere Commerce Payments order message flow completed successfully. WebSphere Commerce Payments can now perform commands on Payments. Legal commands for this state: • CloseOrder, if the order has any payments or credits associated with it, they all must be in closed state before CloseOrder is allowed; if an Order has no payments or credits associated with it, then CloseOrder is not valid. • CancelOrder, if one or the other is true: – The order has no Payments or Credits associated with it, OR – All Payments and Credits are in either Reset, Void, ApprovalExpired or Declined state. • Approve • ApproveReversal • Deposit • DepositReversal
Refundable	The WebSphere Commerce Payments can now perform commands on Payments *and* Credits. The point at which an Order moves from Ordered to Refundable state depends on the payment type. Legal commands for this state: • CloseOrder, if the order has any payments or credits associated with it, they all must be in closed state before CloseOrder is allowed; if an Order has no payments or credits associated with it, then CloseOrder is not valid. • Approve • ApproveReversal • Deposit • DepositReversal • Refund • RefundReversal
Rejected	Indicates that a problem occurred during the consumer-merchant purchase flows. Legal commands for this state: • CancelOrder
Pending	An Order is in Pending state when the WebSphere Commerce Payments is performing a command on the Order. No commands are legal for Orders in this state.
Canceled	This Order has been canceled. Legal commands for this state: • CancelOrder with the DELETEORDER flag enabled (this removes the Order from the database).
Closed	This Order has been closed. Legal commands for this state: • CloseOrder • CancelOrder with the DELETEORDER flag enabled (thisl removes the Order from the database).

Payments

WebSphere Commerce Payments Payment object represents a request by the merchant to the financial institution to approve all or part of an Order.

In many cases, all the money authorized for collection by the Order will be collected in a single payment. Some payment systems may allow the money authorized in one Order (that is, one set of payment instructions) to be collected in multiple payments, depending on the business model. There can be zero or more Payments per Order. The attributes for the Payment object are:

Table 18. PSPaymentObject Attributes

Field name	Syntax	Description
merchantNumber	Numeric token, 1 to 9 digits long	A number that identifies the merchant that created the Order.
orderNumber	Numeric token, 1 to 9 digits long	A number assigned by the merchant that uniquely identifies the Order. This field matches the orderNumber in the Orders table.
paymentNumber	Numeric token, 1 to 9 digits long	A unique identifier for a particular Payment within an Order.
paymentType	Character string	Identifies the payment type, or protocol, used to place the order (for example, SET or OfflineCard).
approvedAmount	Integer	Amount of the Order that has been approved for Payment.
amount	Integer	Identifies the Payment amount in the smallest denomination of the particular currency used to place the order. When combined with AmountExp10, this field specifies the amount of the Payment in the specified currency.
amountExp10	Integer	Indicates the number of decimal places to shift the decimal point to reflect the currency. For example, if the amount is 2325, the currency code is for U.S. dollars, and AmountExp10 is -2, the transaction amount in U.S. dollars is $23.25.
currency	Integer	The currency used to make this Payment. ISO code for currency. For example, 840 is the numeric code for a U.S. dollar, and 392 is the numeric code for a Japanese yen.
timeStampCreated	Date	The time that this Payment entry was created. The number of milliseconds since midnight, January 1, 1970 GMT.
timeStampModified	Date	The time that this Payment entry was last modified. The number of milliseconds since midnight, January 1, 1970 GMT.
state	Character string	The state of the Payment: • payment_reset • payment_approved • payment_deposited • payment_pending • payment_declined • payment_void • payment_closed • payment_approvalexpired
batchNumber	Numeric token, 1 to 9 digits long	The number that identifies the Batch. Assigned when the Payment is deposited.
referenceNumber	Character string	Plain text identifier used by the financial institution to identify a Payment.

Table 18. PSPaymentObject Attributes (continued)

Field name	Syntax	Description
depositAmount	Integer	The amount deposited for this Payment (can differ from approved amount). Assigned when deposited.
merchantAccount	Numeric token, 1 to 9 digits long	A number that identifies the Account used to process this Order.
order	IDREF	XML element representing the order associated with this payment.
approveTime	Date	The last time that this Payment entry was approved.
approvalExpiry	Date	The time that a Payment approval expires. A null value implies no expiration.

Payment states

Payments are in one of the following states:

State	Description	Valid commands
Reset	A Payment enters Reset state when a Payment has been created, but has not yet been processed.	No valid commands exist for Payments in this state, since the Approve command has not yet completed.
Approved	A Payment enters Approved state when an approve command is successful. For credit card payment types, Approved state means that the Payment has been authorized.	• ApproveReversal • Deposit
Deposited	A Payment enters Deposited state when a deposit, or auto-deposit, command is successful. For credit card payment types, Deposited state means that the Payment has been captured.	DepositReversal
Closed	A Payment in Deposited state moves into Closed state when the Batch associated with the Payment closes. When a Payment is in Closed state, the financial transaction is complete; monies are deposited, and the Payment cannot be modified.	No valid commands exist for Payments in this state.
Declined	A Payment enters Declined state when an approve command is rejected for financial reasons.	Approve
Void	A Payment enters Void state when an ApproveReversal command for an amount of zero is successful.	Approve
Pending	A command is currently being performed on this Payment.	No valid commands exist for Payments in this state.

ApprovalExpired	The Payment moves from an Approved state to the ApprovalExpired state after the specified approval time has elapsed or the cassette has detected that the Payment authorization has expired. This is an optional state which may not be supported by a cassette.	ApproveReversal

Split Payments

Suppose a customer contacts an online catalog store and orders $80 of merchandise. The merchant checks the inventory and finds that only $60 worth of merchandise is in stock and can be shipped. The merchant would like to collect $60 now and the remaining $20 when the rest of the order is filled. WebSphere Commerce Payments is designed to support payment systems in which customers provide payment information once (for the entire $80) and the merchant collects the funds over time ($60 now and $20 later). This is referred to as split payments.

AVS common codes

If the cassette you are using supports WebSphere Commerce Payments common AVS codes, then you can also query the **commonAVSCode** parameters to determine the AVS result in a cassette-independent way.

Mapping of common AVS result codes to SET cassette result codes follows.

SET Cassette Result Code	Common AVS Code	PM Constant Name	Description
0	4	AVS_OTHER_RESPONSE	This constant maps the address information unavailable, system unavailable (possibly due to timeout), card type not supported, and transaction ineligible AVS return codes. Some other system-related response was received from the credit card processor.
1	3	AVS_NO_MATCH	Neither the street address nor the postal code matches.
2	2	AVS_POSTALCODE_MATCH	The 5–digit or 9–digit postal code matches, but the street address does not.
3	1	AVS_STREETADDRESS_MATCH	The street address matches, but the postal code does not.
4	0	AVS_COMPLETE_MATCH	This constant maps both the AVS 5–digit and 9–digit postal code and street addresses. Both are exact matches.

Credits

The WebSphere Commerce Payments command that creates the Credit object is called Refund. The Credit object identifies one credit made against the amount of money identified in one Order (that is, the payment agreement) object. There can be zero or more Credits per Order. The attributes for the Credit object are:

Table 19. PSCreditObject Attributes

Field Name	Syntax	Description
merchantNumber	Numeric token, 1 to 9 digits long	A number that identifies the merchant that created the Order.
orderNumber	Numeric token, 1 to 9 digits long	A number assigned by the merchant that uniquely identifies the Order. This field matches the orderNumber in the Orders table.
creditNumber	Numeric token, 1 to 9 digits long	A unique identifier for a particular Credit within an Order.
paymentType	Character string	Identifies the payment type, or protocol, used to place the order (for example, SET or OfflineCard).
amount	Integer	Identifies the Credit amount in the smallest denomination of the particular currency used to place the order. When combined with AmountExp10, this field specifies the amount of the Credit in the specified currency.
amountExp10	Integer	Indicates the number of decimal places to shift the decimal point to reflect the currency. For example, if the amount is 2325, the currency code is for U.S. dollars, and AmountExp10 is -2, the transaction amount in U.S. dollars is $23.25.
currency	Integer	The currency used to issue this Credit. ISO code for currency. For example, 840 is the numeric code for a U.S. dollar, and 392 is the numeric code for a Japanese yen.
timeStampCreated	Date	The time that this Credit entry was created. The number of milliseconds since midnight, January 1, 1970 GMT.
timeStampModified	Date	The time that this Credit entry was last modified. The number of milliseconds since midnight, January 1, 1970 GMT.
state	Character string	The state of the Credit: • credit_reset • credit_refunded • credit_pending • credit_declined • credit_void • credit_closed For more information on Credit states, see “Credit states” on page 119.
batchNumber	Numeric token, 1 to 9 digits long	The number that identifies the Batch. Assigned when the Payment is deposited.
referenceNumber	Character string	Plain text identifier used by the financial institution to identify a Payment.
merchantAccount	Numeric token, 1 to 9 digits long	The number of the Account used to process this Order.

Credit states

Credits are in one of the following states:

State	Description
Reset	A Credit enters Reset state when a Credit has been created, but has not yet been processed. No commands are legal for Credits in this state.
Refunded	A Credit enters Refunded state when a refund command is successful. Legal commands for this state: • RefundReversal
Closed	A Credit in Refunded state moves into Closed state when the Batch associated with the Credit closes. When a Credit is in Closed state, the financial transaction is complete; monies are refunded, and the Credit cannot be modified. No commands are legal for Credits in Closed state.
Declined	A Credit enters Declined state when a refund command is rejected for financial reasons. Legal commands for this state: • Refund
Void	A Credit enters Void state when a RefundReversal command for an amount of zero is successful. Legal commands for this state: • Refund
Pending	A command is currently being performed on this Credit. No commands are legal for Credits in this state.

Batches

A Batch is a collection of financial transactions (Payments and Credits) that are processed as a unit by a financial institution. A Batch is associated with an Account and a merchant. An Account can have zero or more Batches. The attributes for the Batch object are:

Table 20. PSBatchObject Attributes

Field Name	Syntax	Description
merchantNumber	Numeric token, 1 to 9 digits long	The number of the merchant that owns the Batch.
merchantAccount	Numeric token, 1 to 9 digits long	The account number associated with the Batch.
batchNumber	Numeric token, 1 to 9 digits long	The number that identifies the Batch. Assigned when the Payment is deposited.
purgeAllowed	0 or 1 (Boolean)	Flag indicating if it is legal for the merchant to purge this batch. If the value is 1, (*yes*), the merchant can purge this batch using the BatchPurge command. If the value is 0, (*no*), the merchant cannot purge this batch.
forceAllowed	0 or 1 (Boolean)	Flag indicating if it is legal for the merchant to issue a BatchClose command with the Force option set. If the value is 1, (*yes*), the merchant can issue the command.
paymentType	Character string	Identifies the payment type, or protocol, used to place the Order (for example, SET or OfflineCard).

Table 20. PSBatchObject Attributes (continued)

Field Name	Syntax	Description
merchantControl	0 or 1 (Boolean)	Flag indicating if it is legal for the merchant to control this batch. If the value is 1, *(true)*, the merchant is responsible for settling this Batch. (The merchant settles the Batch by explicitly closing the Batch using the BatchClose command.) If the value is 0, *(false)*, the merchant does nothing to settle this Batch.
timeStampOpened	Date	The time that this Batch was opened (either by the merchant or the financial institution). The number of milliseconds since midnight, January 1, 1970 GMT.
timeStampClosed	Date	The time that this Batch was closed (either by the merchant or the financial institution). The number of milliseconds since midnight, January 1, 1970 GMT.
timeStampModified	Date	The time that this Batch was last modified. The number of milliseconds since midnight, January 1, 1970 GMT.
state	Character string	The state of the Batch: • batch_opening • batch_open • batch_closing • batch_closed For more information on Batch states, see “Batch states”.
batchStatus	Character string	The balance status of this Batch: • **batch_not_ yet_ balanced:** balancing has not yet been performed on this Batch. • **batch_balanced:** the Batch has been balanced, and everything is in agreement. • **batch_out_ of_ balance:** the Batch has been balanced, and everything does *not* agree.

Batch states

Batches are in one of the following states:

State	Description
Opening	The Batch is currently being opened. No commands are legal on a Batch in Opening state.
Open	Payments and Credits can be added to a Batch in Open state. Legal commands for this state: • CloseBatch, only if merchantControl is true.
Closing	Batch is currently being settled. No commands are legal for Batches in this state.
Closed	A batch in Closed state has been settled. Legal commands for this state: • DeleteBatch

WebSphere Commerce Payments About objects

WebSphere Commerce Payments defines the following About objects:

- Payment Server About
- Cassette About

Each WebSphere Commerce Payments About object is defined by its attributes, or fields. In the sections that follow, Object Tables display field names, field syntax, and field descriptions for each About object.

Payment Server About

The Payment Server About object contains the version of the WebSphere Commerce Payments. The Payment Server attributes are:

Field Name	Syntax	Description
version	Character string	The WebSphere Commerce Payments version.
userName	Character string	The name of the user running the About command.

Cassette About

The Cassette About object contains version information on a cassette. The Payment Server attributes are:

Field Name	Syntax	Description
cassette	Character string	The cassette payment system name.
version	Character string	The cassette version.

WebSphere Commerce Payments administration objects

WebSphere Commerce Payments defines the following Framework objects for WebSphere Commerce Payments administration:

- Payment Server
- Cassette
- Merchant
- Payment System
- Account
- Event Listener
- User

Each WebSphere Commerce Payments Administration object is defined by its attributes, or fields. In the sections that follow, Object Tables display field names, field syntax, and field descriptions for each Administration object.

Payment Server

The Payment Server object describes the state of the WebSphere Commerce Payments. The Payment Server attributes are:

Table 21. PSPaymentServer Object Attributes

Field Name	Syntax	Description
paymentServerHostname	Character string	The hostname of the computer where WebSphere Commerce Payments is installed.
realmName	Character string	The name of the realm currently being used by the WebSphere Commerce Payments.
logPath	Character string	The directory WebSphere Commerce Payments uses to record trace and error logs. Files that will be written to this directory include `etillerror`, `pserror`, etc. . .
traceFileSize	Integer	The maximum size in bytes of the trace file.
traceSetting	Character string	The current trace setting used by WebSphere Commerce Payments.
numberOfOrderCommands	Integer	The number of order commands made on WebSphere Commerce Payments since the last time it was restarted.
numberOfPaymentCommands	Integer	The number of payment commands made on WebSphere Commerce Payments since the last time it was restarted.
numberOfAdminCommands	Integer	The number of administration commands made on WebSphere Commerce Payments since the last time it was restarted.
numberOfQueryCommands	Integer	The number of query commands made on WebSphere Commerce Payments since the last time it was restarted.
changesPending	Boolean, XML 0 or 1	Flag indicating whether or not changes have been applied to WebSphere Commerce Payments, where 0=false and 1=true. These changes will not take effect until the entire WebSphere Commerce Payments is restarted.
enabled	Boolean, XML 0 or 1	Flag indicating whether WebSphere Commerce Payments is enabled or not (that is, whether WebSphere Commerce Payments is writeable), where 0=false and 1=true.
active	Boolean, XML 0 or 1	Flag indicating whether WebSphere Commerce Payments is active or not (that is, whether WebSphere Commerce Payments is ready for use), where 0=false and 1=true.
valid	Boolean, XML 0 or 1	Flag indicating whether WebSphere Commerce Payments is valid or not (that is, whether WebSphere Commerce Payments is configured correctly), where 0=false and 1=true.
paymentServerMsgs	Character string	A comma-separated list of message codes generated by WebSphere Commerce Payments that identify error, warning, or information messages related to the merchant's Payment settings.

Cassette

The Cassette object describes the state of a cassette that is installed in the WebSphere Commerce Payments. The attributes of a Cassette object are:

Table 22. PSCassetteObject Attributes

Field Name	Syntax	Description
cassette	Character string	The name of the cassette (for example, SET or OfflineCard).
companyPkgName	Character string	The name of the company that developed the cassette (used to identify the cassette's Java package name).
traceSetting	Character string	The current trace setting of the cassette.
changesPending	0 or 1 (Boolean)	Flag indicating whether or not changes have been applied to the cassette, where 0=false and 1=true. These changes will not take effect until the cassette is restarted.
enabled	0 or 1 (Boolean)	Flag indicating whether the cassette is enabled or not (that is, whether the cassette is writeable), where 0=false and 1=true.
active	0 or 1 (Boolean)	Flag indicating whether the cassette is active or not (that is, whether the cassette is ready for use), where 0=false and 1=true.
valid	0 or 1 (Boolean)	Flag indicating whether the cassette is valid or not (that is, whether the cassette is configured correctly), where 0=false and 1=true.
cassetteMsgs	Character string	A comma-separated list of message codes generated by the cassette that identify error, warning, or information messages related to the cassette to the XDM client application.
paymentServerMsgs	Character string	A comma-separated list of message codes generated by WebSphere Commerce Payments that identify error, warning, or information messages related to the cassette.

Merchant

The WebSphere Commerce Payments Merchant object describes the state of a merchant who is defined to use WebSphere Commerce Payments. The attributes of the Merchant are:

Table 23. PSMerchantObject Attributes

Field Name	Syntax	Description
merchantNumber	Numeric token, 1 to 9 digits long	A number that identifies the merchant that created the Order.
merchantName	Character string	The merchant name. This is an optional field that provides meaningful, display information in WebSphere Commerce Payments user interface.
changesPending	0 or 1 (Boolean)	Flag indicating whether or not changes have been applied to the cassette, where 0=false and 1=true. These changes will not take effect until the merchant is re-enabled.
enabled	0 or 1 (Boolean)	Flag indicating whether the cassette is enabled or not, where 0=false and 1=true.
active	0 or 1 (Boolean)	Flag indicating whether the cassette is active or not, where 0=false and 1=true.
valid	0 or 1 (Boolean)	Flag indicating whether the cassette is valid or not, where 0=false and 1=true.

Table 23. PSMerchantObject Attributes (continued)

paymentServerMsgs	Character string	A comma-separated list of message codes generated by WebSphere Commerce Payments, that identify error, warning, or information messages related to the merchant.

Payment System

The WebSphere Commerce Payments Payment System object describes the settings that a merchant has made for a cassette. The attributes of the cassette settings are:

Table 24. PSMerchantCassetteSettingsObject Attributes

Field Name	Syntax	Description
cassette	Character string	The name of the cassette (for example, SET or OfflineCard).
merchantNumber	Numeric token, 1 to 9 digits long	A number that identifies the merchant.
changesPending	0 or 1 (Boolean)	Flag indicating whether or not changes have been applied to the cassette, where 0=false and 1=true. These changes will not take effect until the cassette is restarted for this merchant.
enabled	0 or 1 (Boolean)	Flag indicating whether the cassette is enabled or not (that is, whether the cassette is writeable), where 0=false and 1=true.
active	0 or 1 (Boolean)	Flag indicating whether the cassette is active or not (that is, whether the cassette is ready for use), where 0=false and 1=true.
valid	0 or 1 (Boolean)	Flag indicating whether the cassette is valid or not (that is, whether the cassette is configured correctly), where 0=false and 1=true.
paymentServerMsgs	Character string	A comma-separated list of message codes generated by WebSphere Commerce Payments that identify error, warning, or information messages related to the payment system.

Account

The WebSphere Commerce Payments merchant Account object describes the state of an account that a merchant holds with a financial institution. The attributes of an account are:

Table 25. PSMerchantAccountObject Attributes

Field Name	Syntax	Description
cassette	Character string	The name of the cassette (for example, SET or OfflineCard).
merchantNumber	Numeric token, 1 to 9 digits long	A number that identifies the merchant.
merchantAccountNumber	Numeric token, 1 to 9 digits long	A number that identifies the account. This number is created locally (that is, by the hosting service provider or by the merchant administrator) and is for tracking purposes.
merchantAccountName	Character string	The account name. This is an optional field that provides meaningful, display information in the Payment Manger user interface.

Table 25. PSMerchantAccountObject Attributes (continued)

Field Name	Syntax	Description
financialInstName	Character string	The financial institution name. This is an optional field that provides meaningful, display information in WebSphere Commerce Payments user interface.
changesPending	0 or 1 (Boolean)	Flag indicating whether or not changes have been applied to the cassette, where 0=false and 1=true. These changes will not take effect until the account is restarted.
enabled	0 or 1 (Boolean)	Flag indicating whether the cassette is enabled or not, where 0=false and 1=true.
active	0 or 1 (Boolean)	Flag indicating whether the cassette is active or not, where 0=false and 1=true.
valid	0 or 1 (Boolean)	Flag indicating whether the cassette is valid or not, where 0=false and 1=true.
cassetteMsgs	Character string	A comma-separated list of message codes generated by the cassette that identify error, warning, or information messages related to the account or the XDM client application.
paymentServerMsgs	Character string	A comma-separated list of message codes generated by the Payment Manger that identify error, warning, or information messages related to the account.
apApproveFlag	Numeric Token, 1 to 9 digits long	Approve flag for AcceptPayment
apDepositFlag	0 or 1 (Boolean)	0=false and 1=true. Deposit flag forAcceptPayment. Should only be specified when apApproveFlag is defined and not set to 0.
rpApproveFlag	Numeric Token, 1 to 9 digits long	Approve flag for ReceivePayment
rpDepositFlag	0 or 1 (Boolean)	0=false and 1=true. Deposit flag for ReceivePayment. Should only be specified when rpApproveFlag is defined and not set to 0.
approvalExpiration	Numeric token, 1 to 9 digits long	Value indicating the number of days from the time a payment is approved until the payment approval expires.

Event Listener

The WebSphere Commerce Payments Event Listener object describes the state of registered WebSphere Commerce Payments events. The attributes of an Event Listener are:

Table 26. PSEventListenerObject Attributes

Field Name	Syntax	Description
eventType	Character string	The type of event being monitored.
listenerURL	Character string	The URL defined for an event type. WebSphere Commerce Payments event notification model provides for messages to be sent to the listener URL defined for a specific event type. Multiple URLs can be defined for a single event type.
timeRegistered	Date	The time that the merchant registered an event type. The number of milliseconds since midnight, January 1, 1970 GMT.
socksHost	Character string	The hostname of the socks server receiving event notification from WebSphere Commerce Payments. The value is `null` if not using socks. The default is `null`.

Table 26. PSEventListenerObject Attributes (continued)

Field Name	Syntax	Description
socksPort	Character string	The port of the socks server receiving event notification from WebSphere Commerce Payments. The value is `null` if not using socks. The default is `null`.
merchantNumber	Numeric token, 1 to 9 digits long	A number that identifies the merchant.
changesPending	0 or 1 (Boolean)	Flag indicating whether or not changes have been applied to the cassette, where 0=false and 1=true. These changes will not take effect until the cassette is restarted. Not used.
enabled	0 or 1 (Boolean)	Flag indicating whether the cassette is enabled or not, where 0=false and 1=true.
active	0 or 1 (Boolean)	Flag indicating whether the cassette is active or not, where 0=false and 1=true. Not used.
valid	0 or 1 (Boolean)	Flag indicating whether the cassette is valid or not, where 0=false and 1=true. Not used
paymentServerMsgs	Character string	A comma-separated list of message codes generated by WebSphere Commerce Payments that identify error, warning, or information messages related to the event type.

User

The WebSphere Commerce Payments User object describes the state of users defined for the WebSphere Commerce Payments. The attributes of a User are:

Table 27. PSUserObject Attributes

Field Name	Syntax	Description
userName	Character string	The name of the user.
configuration	Character string	The user configuration.
roleIDs	Character string.	The role ID defined for the user (that is, clerk, supervisor, Merchant administrator, or Payments administrator).
merchantNumber	Numeric token, 1 to 9 digits long	A number that identifies the merchant. This value is set for all roles *other than* Payments administrator. Note that the result of the QueryUsers command may return a user with access rights to multiple merchants. In this case, the WebSphere Commerce Payments will return the merchant number as a list of merchant numbers with the following syntax: `m1, m2, m3, . . .`
changesPending	0 or 1 (Boolean)	Flag indicating whether or not changes have been applied to the User, where 0=false and 1=true. These changes will not take effect until the cassette is restarted for the merchant. Not used.
enabled	0 or 1 (Boolean)	Flag indicating whether the cassette is enabled or not, where 0=false and 1=true (enabled).
active	0 or 1 (Boolean)	Flag indicating whether the cassette is active or not, where 0=false and 1=true. Not used.
valid	0 or 1 (Boolean)	Flag indicating whether the cassette is valid or not, where 0=false and 1=true. Not used.
paymentServerMsgs	Character string	A comma-separated list of message codes generated by WebSphere Commerce Payments that identify error, warning, or information messages related to the user.
objectCount		The number of real, matched objects.

Part 4. Appendixes

Appendix A. WebSphere Commerce Payments return codes

The return codes include both *primary return codes* and *secondary return codes*.

- Primary return codes (PRCs) describe the basic response of WebSphere Commerce Payments. The primary return code is returned on each command.
- Secondary return codes (SRCs) provide additional information. WebSphere Commerce Payments defines two types of generic SRCs: a set that is common to all the PRCs, and a set that is specific to a particular PRC.

  The SRC is returned in the optional secondaryrc structure passed on each command.

Protocol cassette writers may also extend the set with protocol-specific codes. Refer to the appropriate cassette supplement for information regarding these codes.

Primary return codes (PRCs)

The following table shows the primary return codes (PRCs) for WebSphere Commerce Payments. Those PRCs that have specific secondary return codes (SRCs) are listed in this table; SRCs that span multiple PRCs are in “Secondary return codes (generic)” on page 131.

Table 28. Primary Return Codes (PRCs)

PRC_OPERATION_SUCCESS	0	Operation completed successfully. A non-zero secondary return code (SRC) may be provided for additional information.
PRC_OPERATION_PENDING	1	The API call has not yet completed and is pending on the availability of WebSphere Commerce Payments entities. The SRC indicates resources upon which the operation is pending.
PRC_UNDEFINED_OBJECT	2	A specified object was not found. The object is indicated by the SRC.
PRC_PARAMETER_NOT_FOUND	3	A required parameter was not found. The parameter is indicated by the SRC.
PRC_PARAMETER_TOO_SHORT	4	A required parameter was too short. The parameter is indicated by the SRC.
PRC_PARAMETER_TOO_LONG	5	A required parameter was too long. The parameter is indicated by the SRC.
PRC_PARAMETER_FORMAT_ERROR	6	A required parameter was formatted incorrectly. The parameter is indicated by the SRC.
PRC_PARAMETER_VALUE_ERROR	7	A required parameter had an incorrect value. The parameter is indicated by the SRC.
PRC_DUPLICATE_OBJECT	8	A duplicate object exists. As indicated by the SRC, a payment with this payment number already exists.
PRC_PARAMETER_MISMATCH	9	A parameter mismatch occurred. The parameter is indicated by the SRC.

Table 28. Primary Return Codes (PRCs) (continued)

PRC_INPUT_ERROR	10	There was an error parsing the input stream. The command or one of its parameters has an invalid length.
PRC_VERB_NOT_VALID_IN_PRESENT_STATE	11	An object is not in the correct state for this action. The particular object is indicated by the SRC.
PRC_COMMUNICATION_ERROR	12	A communication error occurred in the WebSphere Commerce Payments.
PRC_INTERNAL_ETILL_ERROR	13	The WebSphere Commerce Payments experienced an unexpected internal error.
PRC_DATABASE_ERROR	14	A database communications error occurred.
PRC_CASSETTE_ERROR	15	A cassette-specific error occurred. Refer to cassette-supplementary information for documentation.
PRC_UNSUPPORTED_API_VERSION	17	The API version used by the application program is newer than that supported by the WebSphere Commerce Payments.
PRC_OBSOLETE_API_VERSION	18	The API version used by the application is no longer supported by the WebSphere Commerce Payments. Upgrade the application program to use the newer function which replaces the obsoleted function or feature.
PRC_AUTOAPPROVE_FAILED	19	Auto approve in ReceivePayment or AcceptPayment failed.
PRC_AUTODEPOSIT_FAILED	20	Auto deposit in ReceivePayment or AcceptPayment failed
PRC_CASSETTE_NOTRUNNING	21	The cassette is not running.
PRC_CASSETTE_NOTVALID	22	The cassette is not valid.
PRC_UNSUPPORTED_IN_SYSPLEX	23	The operation is not supported in sysplex environment.
PRC_PARAMETER_NULL_VALUE	24	The parameter has a null value.
PRC_XML_ERROR	30	The XML document is not correct.
PRC_COREQUISITE_PARAMETER_NOT_FOUND	31	The parameter must be specified when another parameter is specified.
PRC_INVALID_PARAMETER_COMBINATION	32	The combination of the parameters specified in a API command is not allowed.
PRC_BATCH_ERROR	33	An error related with the Batch operation occurred.
PRC_FINANCIAL_FAILURE	34	The operation failed for financial reasons.
PRC_SERVLET_INIT_ERROR	50	An error occurred when initializing the servlet.
PRC_AUTHENTICATION_ERROR	51	An error occurred during the user authentication.

Table 28. Primary Return Codes (PRCs) (continued)

PRC_AUTHORIZATION_ERROR	52	An error occurred during the user authorization.
PRC_UNHANDLED_EXCEPTION	53	An unhandled (such as null pointer) exception occurred.
PRC_DUPLICATE_PARAMETER_VALUE_NOT_ALLOWED	54	The parameter can not be specified multiple times in this API command.
PRC_COMMAND_NOT_SUPPORTED	55	The command name is not recognized as a valid $til; command.
PRC_CRYPTO_ERROR	56	Error related with encryption/decryption key.
PRC_NOT_ACTIVE	57	An administration object is not active.
PRC_PARAMETER_NOT_ALLOWED	58	The parameter should not be specified.
PRC_DELETE_ERROR	59	The object could not be deleted.
PRC_WEBSPHERE	60	A WebSphere/WebServer related error occurred.
PRC_SUPPORTED_IN_SYSPLEX_ADMIN_ONLY	61	The request is supported in Sysplex mode only on the WebSphere Commerce Payments designated as the Sysplex Administrator.
PRC_REALM	62	A realm related error occurred.

Secondary return codes (generic)

Table 29. Generic Secondary Return Codes (SRCs)

RC_NONE	0	No additional information available.
RC_INITIALIZATION_MESSAGE	1	An initialization message is included in the return data buffer. This buffer must be freed by the caller of this routine.
RC_INPUT_ERROR_TOO_LONG	2	Input stream exceeds maximum length.
RC_INPUT_ERROR_UNKNOWN_COMMAND	3	Unknown command.
RC_UNEXPECTED	4	An unexpected error has occurred.
RC_COMMUNICATION_ERROR_INPUT	5	WebSphere Commerce Payments received an exception when reading data from the merchant server.
RC_API_INITIALIZE_FAILURE	6	API initialization failed.
RC_MERCHANTNUMBER	110	Response refers to the merchant number parameter.
RC_ORDERNUMBER	111	Response refers to the order number parameter.
RC_PAYMENTNUMBER	112	Response refers to the PAYMENTNUMBER parameter.
RC_CREDITNUMBER	113	Response refers to the CREDITNUMBER parameter.
RC_BATCHNUMBER	114	Response refers to the BATCHNUMBER parameter. (Note: In previous versions this return code referenced the BATCHID parameter.)

Table 29. Generic Secondary Return Codes (SRCs) (continued)

RC_ACCOUNTNUMBER	115	Response refers to the ACCOUNTNUMBER parameter.
RC_PAYMENTTYPE	116	Response refers to the PAYMENTTYPE parameter.
RC_AMOUNT	117	Response refers to the AMOUNT parameter.
RC_AMOUNTEXP10	118	Response refers to the AMOUNTEXP10 parameter.
RC_CURRENCY	119	Response refers to the CURRENCY parameter.
RC_OD	120	Response refers to the order description parameter.
RC_CHARSET	121	Response refers to the character set parameter.
RC_SUCCESSURL	122	Response refers to the success URL parameter.
RC_FAILURL	123	Response refers to the failure URL parameter.
RC_CANCELURL	124	Response refers to the cancel URL parameter.
RC_APPROVEFLAG	125	Response refers to the approve flag parameter.
RC_PAYMENTAMOUNT	126	Response refers to the payment amount parameter.
RC_SPLITFLAG	127	Response refers to the splits allowed parameter.
RC_DEPOSITFLAG	128	Response refers to the deposit flag parameter.
RC_PROTOCOLDATA	129	Response refers to the protocol data parameter.
RC_ORDERURLS	130	Response refers to the order URL parameter.
RC_SERVICEURL	131	Response refers to the service URL parameter.
RC_CASSETTECOMMAND	132	Response refers to the cassette command parameter.
RC_USERNAME	133	Response refers to the user parameter.
RC_EVENTTYPE	134	Response refers to the event type parameter.
RC_WITHCREDITS	135	Response refers to the withCredits parameter.
RC_CREATEBEGINTIME	136	Response refers to the creation begin time parameter.
RC_CREATEENDTIME	137	Response refers to the creation end time parameter.
RC_MINAMOUNT	138	Response refers to the minimum amount parameter.

Table 29. Generic Secondary Return Codes (SRCs) (continued)

RC_MAXAMOUNT	139	Response refers to the maximum amount parameter.
RC_RETURNATMOST	140	Response refers to the ″return at most″ parameter.
RC_KEYSONLY	141	Response refers to the keys only parameter.
RC_DTDPATH	143	Response refers to the dtd path parameter.
RC_REFERENCENUMBER	144	Response refers to the reference number parameter.
RC_WITHORDERS	145	Response refers to the withOrders parameter.
RC_MESSAGES	146	Response refers to the messages key.
RC_OPENBEGINTIME	147	Response refers to the batch open beginning time parameter.
RC_OPENENDTIME	148	Response refers to the batch open ending time parameter.
RC_CLOSEBEGINTIME	149	Response refers to the batch close beginning time parameter.
RC_CLOSEENDTIME	150	Response refers to the batch close ending time parameter.
RC_STATUS	151	Response refers to the status parameter.
RC_CLOSEALLOWED	153	Response refers to the close allowed parameter.
RC_WITHPAYMENTS	154	Response refers to the withPayments parameter.
RC_TIMEREGISTERED	155	Response refers to the time registered parameter.
RC_MINAPPROVEAMOUNT	156	Response refers to the minimum approve amount parameter.
RC_MAXAPPROVEAMOUNT	157	Response refers to the maximum approve amount parameter.
RC_MINDEPOSITAMOUNT	158	Response refers to the minimum deposit amount parameter.
RC_MAXDEPOSITAMOUNT	159	Response refers to the maximum deposit amount parameter.
RC_ORDERURL	160	Response refers to the order URL parameter.
RC_MODIFYBEGINTIME	161	Response refers to the modification beginning time parameter.
RC_MODIFYENDTIME	162	Response refers to the modification ending time parameter.
RC_DELETEORDER	165	Response refers to the delete order parameter.
RC_MINUNAPPROVEDAMOUNT	166	Response refers to the minimum un-approved amount parameter.

Table 29. Generic Secondary Return Codes (SRCs) (continued)

RC_MAXUNAPPROVEDAMOUNT	167	Response refers to the maximum un-approved amount parameter.
RC_APPROVESALLOWED	168	Response refers to the approve allowed parameter.
RC_PURGEALLOWED	169	Response refers to the PURGEALLOWED parameter.
RC_MAXBATCHSIZE	170	Response refers to the $MAXBATCHSIZE parameter.
RC_CHECK_CASSETTE_STATUS	171	Inspect cassette-specific data for further information.
RC_FORCE	172	Response refers to the FORCE parameter. May be returned in response to the BATCHCLOSE command. Indicates that the error described by the primary return code refers to the boolean parameter FORCE.
RC_AP_APPROVEFLAG	173	Response refers to the acceptPayment approve flag parameter.
RC_AP_DEPOSITFLAG	174	Response refers to the acceptPayment deposit flag parameter.
RC_RP_APPROVEFLAG	175	Response refers to the receivePayment approve flag parameter.
RC_RP_DEPOSITFLAG	176	Response refers to the receivePayment deposit flag parameter.
RC_APPROVALEXPIRATION	177	Response refers to the ApprovalExpiration parameter.
RC_MERCHANTPAYSYS	202	Response refers to merchant payment system (such as SET).
RC_ACCOUNT	203	Response refers to an account.
RC_ORDER	204	Response refers to an order entity.
RC_PAYMENT	205	Response refers to a payment entity.
RC_CREDIT	206	Response refers to a credit entity.
RC_BATCH	207	Response refers to a batch entity.
RC_BRAND	208	Response refers to a brand.
RC_STATE	209	Response refers to the state.
RC_MULTIPLE_BATCHES	211	Response refers to batch objects.
RC_AUTOMATIC_CREATION	212	An error occurred during automatic batch open
RC_BATCH_EMPTY	213	The batch is empty. An attempt was made to close a batch that does not contain any payments or credits. It is up to the cassette to decide whether or not this is an error condition.
RC_SYSPLEXFLAG	214	Response refers to the sysplex flag.
RC_COMMTYPE	215	Response refers to the communication type.

Table 29. Generic Secondary Return Codes (SRCs) (continued)

RC_PAYMENTGROUPNAME	216	Response refers to the payment group name.
RC_ADMINHOSTNAME	217	Response refers to the admin host name.
RC_NDHOSTNAME	218	Response refers to the Net.Dispatcher host name.
RC_PLEXNAME	219	Response refers to the sysplex name.
RC_UNKNOWN_ETILL_HOST	301	The specified WebSphere Commerce Payments host is not valid.
RC_HOSTNAME_NOT_VALID	303	The WebSphere Commerce Payments hostname parameter is in error.
RC_HOST_IP_ADDRESS_UNAVAILABLE	306	Could not locate host IP address.
RC_SOCKET_STARTUP_FAILURE	307	Could not initialize socket library.
RC_HANDLE_REQUIRED	308	A PaymentServerHandle is required for this API.
RC_COMMUNICATION_ERROR	309	A communication error occurred.
RC_RESERVED_BITS_SET_IN_FLAGS	310	Bits that are reserved for future use are non-zero. They must be zero.
RC_TIME_PERIOD_INVALID	311	The value specified on the TimePeriod is invalid.
RC_PROTOCOL_DATA_KEYWORD_INVALID	312	The keyword in the protocol data is not valid.
RC_AMOUNT_RANGE_INVALID	313	The amount range is not valid.
RC_SOCKET_CREATION_FAILED	320	Could not open a socket to communicate with the WebSphere Commerce Payments. TCP/IP socket resources may be depleted.
RC_CONNECTION_TO_PAYMENT_SERVER_FAILED	321	Could not open a network connection to the WebSphere Commerce Payments using port and address specified earlier on an etInitializeAPI() call.
RC_SEND_OF_DATA_ON_SOCKET_FAILED	322	Could not send data on network connection with WebSphere Commerce Payments. WebSphere Commerce Payments may have closed the connection.
RC_RECEIVE_OF_DATA_ON_SOCKET_FAILED	323	Could not receive data on network connection with WebSphere Commerce Payments. WebSphere Commerce Payments may have closed the connection.
RC_ERROR_CHECKING_FOR_READ_DATA	324	Could not check for data ready to read on network connection with WebSphere Commerce Payments. WebSphere Commerce Payments may have closed the connection.
RC_SOCKET_CLOSE_FAILED	325	Failed to close the socket.
RC_ENCODING_EXCEPTION	400	An encoding error occurred.

Table 29. Generic Secondary Return Codes (SRCs) (continued)

RC_UNSUPPORTED_DOCUMENT_TYPE	401	The XML document type is not supported.
RC_EMPTY_DOCUMENT	402	The document is empty.
RC_MISSING_ORDER_COLLECTION	403	The order collection is missing.
RC_DOCUMENT_TOO_LARGE	404	The XML document generated by an XDM query was too large. Refine the search criteria and re-attempt the query.
RC_SERVLET_INIT_EXCEPTION	500	An error occurred during the servlet initialization.
RC_CANNOT_FIND_PROPERTY_FILE	501	The property file can not be located.
RC_ERROR_LOADING_PROPERTY_FILE	502	An error occurred while loading the property file.
RC_ERROR_JDBCDRIVER_NAME	503	Response refers to the JDBC driver name.
RC_ERROR_JDBCURL	504	Response refers to the JDBC URL.
RC_ERROR_DBOWNER	505	Response refers to the database owner.
RC_ERROR_DBUSERID	506	Response refers to the database user id.
RC_ERROR_DBPASSWORD	507	Response refers to the database password.
RC_ERROR_LOGPATH	508	Response refers to the log path.
RC_ERROR_HOSTNAME	509	Response refers to the host name.
RC_ERROR_PSENGINE_PORTNUMBER	510	Response refers to the WebSphere Commerce Payments engine port number.
RC_ERROR_LOADING_JDBCDRIVER	511	An error occurred while loading JDBC driver.
RC_ERROR_CONNECTING_DATABASE_OR_EXEC_SQL	512	An error occurred while either connecting to the database or executing the SQL statement.
RC_ERROR_INIT_ERROR_LOG	513	An error occurred while initializing the error log.
RC_ERROR_LOADING_CASSETTE	514	An error occurred while loading the cassette.
RC_ERROR_ROOT_PASSWORD	515	The root password is not valid.
RC_ERROR_MAXDBCONNECTIONS	516	Response refers to the maximum number of database connections.
RC_ERROR_MINSENSITIVEACCESSROLE	517	Response refers to the minimum role allowed to view sensitive financial data.
RC_NEW_PASSWORD	518	Parameter refers to the new password.
RC_DATA_SOURCE	519	Parameter refers to the data source name.
RC_OPERATION	530	Response refers to the Operation parameter.
RC_ETAPIVERSION	531	Response refers to the etApiVersion parameter.

Table 29. Generic Secondary Return Codes (SRCs) (continued)

RC_AUTHENTICATED_USER_NOT_GIVEN	553	No authenticated user was given for the WebSphere Commerce Payments command.
RC_USER_NOT_AUTHORIZED	554	The specified user is not authorized to perform the requested operation.
RC_ERROR_PROTECTION_REALM_NOT_SPECIFIED	555	There is no name specified for the ProtectedRealm setting in the PaymentServlet.properties file.
RC_SPECIFIED_REALM_UNKNOWN	556	The realm specified in the PaymentServlet.properties file is unknown.
RC_REALMCLASS	557	Response refers to the eTill.RealmClass property.
RC_PAYSERVER_ADMIN	600	Response refers to the WebSphere Commerce Payments administration entity.
RC_CASSETTE_ADMIN	601	Response refers to a cassette administration entity.
RC_MERCHANT_ADMIN	602	Response refers to a merchant administration entity.
RC_PAYMENTSYSTEM_ADMIN	603	Response refers to a payment system administration entity.
RC_ACCOUNT_ADMIN	604	Response refers to an account administration entity.
RC_ETILLHOSTNAME	611	Response refers to the ETILLHOSTNAME parameter.
RC_TRACESETTING	612	Response refers to the TRACESETTING parameter.
RC_TRACEFILESIZE	613	Response refers to the TRACEFILESIZE parameter.
RC_LOGPATH	614	Response refers to the LOGPATH parameter.
RC_CASSETTENAME	615	Response refers to the CASSETTENAME parameter.
RC_MERCHANTTITLE	616	Response refers to the MERCHANTTITLE parameter.
RC_ACCOUNTTITLE	617	Response refers to the ACCOUNTTITLE parameter.
RC_FINANCIALINSTITUTION	618	Response refers to the FINANCIALINSTITUTION parameter.
RC_OBJECTNAME	619	Response refers to the OBJECTNAME parameter.
RC_ENABLED	620	Response refers to the ENABLED parameter.
RC_EVENT_LISTENER	621	Response refers to the EVENTLISTENER object.
RC_LISTENERURL	622	Response refers to the LISTENERURL parameter.

Table 29. Generic Secondary Return Codes (SRCs) (continued)

RC_SOCKSPORT	623	Response refers to the SOCKSPORT parameter.
RC_ROLE	624	Response refers to the user role parameter.
RC_USER	625	Response refers to the user object.
RC_USER_NOT_ENABLED	626	Response refers to the user (the user is not enabled).
RC_USER_MISCONFIGURED	627	Response refers to the User object (the user has rights to the WebSphere Commerce Payments. The user is misconfigured).
RC_KEY_TAMPERED	628	Encryption key has been altered.
RC_KEY_NOT_EXIST	629	Encryption key did not exist for the specified component.
RC_SOCKSHOST	630	Response refers to the SOCKSHOST parameter.
RC_ENCRYPT_ENCRYPTION_KEY_FAILED	631	Failed to encrypt encryption key.
RC_DECRYPT_ENCRYPTION_KEY_FAILED	632	Failed to decrypt encryption key.
RC_ENCRYPTION_KEY_TYPE_NOT_SUPPORTED	633	The encryption key type is not supported.
RC_VALIDATE_ENCRYPTION_KEY_FAILED	634	Failed to validate encryption key.
RC_GENERATE_ENCRYPTION_KEY_FAILED	635	Failed to generate encryption key.
RC_NOT_ACL_OWNER	636	The user is not the ACL owner.
RC_BAD_REALM	637	A realm error has been occurred.
RC_NO_SUCH_ACL	638	The ACL is not defined.
RC_LAST_ACL_OWNER	639	The user is the last owner of the ACL.
RC_NO_SUCH_USER	640	The user is not defined in the WebSphere realm.
RC_USER_BEING_REMOVED_HAS_NO_ACCESS_RIGHTS	641	Try to remove a user's access rights who does not have one in the WebSphere Commerce Payments.
RC_FILTER	642	Response refers to the FILTER parameter.
RC_TRANSACTIONID	643	Response refers to the TRANSACTIONID parameter.
RC_ORDERDATA1	644	Response refers to the ORDERDATA1 parameter.
RC_ORDERDATA2	645	Response refers to the ORDERDATA2 parameter.
RC_ORDERDATA3	646	Response refers to the ORDERDATA3 parameter.
RC_ORDERDATA4	647	Response refers to the ORDERDATA4 parameter.
RC_ORDERDATA5	648	Response refers to the ORDERDATA5 parameter.
RC_SERVICE_POOL	649	Response refers to the service thread pool size eTill.spoolsize.

Table 29. Generic Secondary Return Codes (SRCs) (continued)

RC_INVALID_CHANGEPASSWORD_STATE	650	It is only valid to change the PM password immediately after the WebSphere Commerce Payments Application Server is started.
RC_ASYNCHAPPDELAY	651	Response refers to the wpm.AsynApproveDelayTimeInSecs parameter.
RC_APPEXPDELAY	652	Response refers to the wpm.ApprovalExpirationDelayTimeInMins parameter.
RC_PROTOCOL_POOL	653	Response refers to the protocol thread pool size wpm.ppoolsize.
RC_CASSETTE_PCARD_SHIPPINGAMOUNT	900	Response refers to purchase card data's shipping amount parameter.
RC_CASSETTE_PCARD_DUTYAMOUNT	901	Response refers to purchase card data's duty amount parameter.
RC_CASSETTE_PCARD_DUTYREFERENCE	902	Response refers to purchase card data's duty reference parameter.
RC_CASSETTE_PCARD_NATIONALTAXAMOUNT	903	Response refers to purchase card data's national tax amount parameter.
RC_CASSETTE_PCARD_NATIONALTAXRATE	904	Response refers to purchase card data's national tax rate parameter.
RC_CASSETTE_PCARD_LOCALTAXAMOUNT	905	Response refers to purchase card data's local tax amount parameter.
RC_CASSETTE_PCARD_OTHERTAXAMOUNT	906	Response refers to purchase card data's other tax amount parameter.
RC_CASSETTE_PCARD_TOTALTAXAMOUNT	907	Response refers to purchase card data's total tax amount parameter.
RC_CASSETTE_PCARD_MERCHANTTAXID	908	Response refers to purchase card data's merchant tax id parameter.
RC_CASSETTE_PCARD_ALTERNATETAXID	909	Response refers to purchase card data's alternate tax id parameter.
RC_CASSETTE_PCARD_TAXEXEMPTINDICATOR	910	Response refers to purchase card data's tax exempt indicator parameter.
RC_CASSETTE_PCARD_MERCHANTDUTYTARIFFREFERENCE	911	Response refers to purchase card data's merchant duty tariff reference parameter.
RC_CASSETTE_PCARD_CUSTOMERDUTYTARIFFREFERENCE	912	Response refers to purchase card data's customer duty tariff reference parameter.
RC_CASSETTE_PCARD_SUMMARYCOMMODITYCODE	913	Response refers to purchase card data's summary commodity code parameter.
RC_CASSETTE_PCARD_MERCHANTTYPE	914	Response refers to purchase card data's merchant type parameter.
RC_CASSETTE_PCARD_MERCHANTCOUNTRYCODE	915	Response refers to purchase card data's merchant country code parameter.
RC_CASSETTE_PCARD_MERCHANTCITYCODE	916	Response refers to purchase card data's merchant city code parameter.

Table 29. Generic Secondary Return Codes (SRCs) (continued)

RC_CASSETTE_PCARD_MERCHANTSTATEPROVINCE	917	Response refers to purchase card data's merchant state province parameter.
RC_CASSETTE_PCARD_MERCHANTPOSTALCODE	918	Response refers to purchase card data's merchant postal code parameter.
RC_CASSETTE_PCARD_MERCHANTLOCATIONID	919	Response refers to purchase card data's merchant location id parameter.
RC_CASSETTE_PCARD_MERCHANTNAME	920	Response refers to purchase card data's merchant name parameter.
RC_CASSETTE_PCARD_SHIPFROMCOUNTRYCODE	921	Response refers to purchase card data's ship from country code parameter.
RC_CASSETTE_PCARD_SHIPFROMCITYCODE	922	Response refers to purchase card data's ship from city code parameter.
RC_CASSETTE_PCARD_SHIPFROMSTATEPROVINCE	923	Response refers to purchase card data's ship from state province parameter.
RC_CASSETTE_PCARD_SHIPFROMPOSTALCODE	924	Response refers to purchase card data's ship from postal code parameter.
RC_CASSETTE_PCARD_SHIPFROMLOCATIONID	925	Response refers to purchase card data's ship from location id parameter.
RC_CASSETTE_PCARD_SHIPTOCOUNTRYCODE	926	Response refers to purchase card data's ship to country code parameter.
RC_CASSETTE_PCARD_SHIPTOCITYCODE	927	Response refers to purchase card data's ship to city code parameter.
RC_CASSETTE_PCARD_SHIPTOSTATEPROVINCE	928	Response refers to purchase card data's ship to state province parameter.
RC_CASSETTE_PCARD_SHIPTOPOSTALCODE	929	Response refers to purchase card data's ship to postal code parameter.
RC_CASSETTE_PCARD_SHIPTOLOCATIONID	930	Response refers to purchase card data's ship to location id parameter.
RC_CASSETTE_PCARD_MERCHANTORDERNUMBER	931	Response refers to purchase card data's merchant order number parameter.
RC_CASSETTE_PCARD_CUSTOMERREFERENCENUMBER	932	Response refers to purchase card data's customer reference number parameter.
RC_CASSETTE_PCARD_ORDERSUMMARY	933	Response refers to purchase card data's order summary parameter.
RC_CASSETTE_PCARD_CUSTOMERSERVICEPHONE	934	Response refers to purchase card data's customer service phone parameter.
RC_CASSETTE_PCARD_DISCOUNTAMOUNT	935	Response refers to purchase card data's discount amount parameter.
RC_CASSETTE_PCARD_SHIPPINGNATIONALTAXRATE	936	Response refers to purchase card data's shipping national tax rate parameter.
RC_CASSETTE_PCARD_SHIPPINGNATIONALTAXAMOUNT	937	Response refers to purchase card data's shipping national tax amount parameter.

Table 29. Generic Secondary Return Codes (SRCs) (continued)

RC_CASSETTE_PCARD_NATIONALTAXINVOICEREFERENCE	938	Response refers to purchase card data's national tax invoice reference parameter.
RC_CASSETTE_PCARD_PRINTCUSTOMERSERVICEPHONENUMBER	939	Response refers to purchase card data's print customer service phone number parameter.
RC_CASSETTE_ITEM_COMMODITYCODE	940	Response refers to line item data's commodity code parameter.
RC_CASSETTE_ITEM_PRODUCTCODE	941	Response refers to line item data's product code parameter.
RC_CASSETTE_ITEM_DESCRIPTOR	942	Response refers to line item data's descriptor parameter.
RC_CASSETTE_ITEM_QUANTITY	943	Response refers to line item data's quantity parameter.
RC_CASSETTE_ITEM_SKU	944	Response refers to line item data's SKU parameter.
RC_CASSETTE_ITEM_UNITCOST	945	Response refers to line item data's unit cost parameter.
RC_CASSETTE_ITEM_UNITOFMEASURE	946	Response refers to line item data's unit of measure parameter.
RC_CASSETTE_ITEM_NETCOST	947	Response refers to line item data's net cost parameter.
RC_CASSETTE_ITEM_DISCOUNTAMOUNT	948	Response refers to line item data's discount amount parameter.
RC_CASSETTE_ITEM_DISCOUNTINDICATOR	949	Response refers to line item data's discount indicator parameter.
RC_CASSETTE_ITEM_NATIONALTAXAMOUNT	950	Response refers to line item data's national tax amount parameter.
RC_CASSETTE_ITEM_NATIONALTAXRATE	951	Response refers to line item data's national tax rate parameter.
RC_CASSETTE_ITEM_NATIONALTAXTYPE	952	Response refers to line item data's national tax type parameter.
RC_CASSETTE_ITEM_LOCALTAXAMOUNT	953	Response refers to line item data's local tax amount parameter.
RC_CASSETTE_ITEM_LOCALTAXRATE	954	Response refers to line item data's local tax rate parameter.
RC_CASSETTE_ITEM_OTHERTAXAMOUNT	955	Response refers to line item data's other tax amount parameter.
RC_CASSETTE_ITEM_TOTALCOST	956	Response refers to line item data's total cost parameter.
RC_CASSETTE_FUNCTION_NOT_SUPPORTED	1000	The cassette does not support this command.
RC_CASSETTE_UNSPECIFIED_ERROR	1001	The cassette does not support this command.
RC_CASSETTE_BATCH_ID	1002	Batch ID was either (1) specified when prohibited or (2) not specified when required.

Table 29. Generic Secondary Return Codes (SRCs) (continued)

RC_CASSETTE_REFUND_AMOUNT_NOT_ZERO	1003	The cassette allows only complete refund reversals (that is, the amount must be zero).
RC_CASSETTE_OPERATION_FAILED	1004	The operation experienced financial failure.
RC_CASSETTE_ENCRYPTION_ERROR	1008	An encryption error occurred while the cassette was composing or processing a protocol message.
RC_CASSETTE_DECRYPTION_ERROR	1009	A decryption error occurred while the cassette was composing or processing a protocol message.
RC_CASSETTE_IMPLICIT_BATCHES_ONLY	1010	A BATCHOPEN or BATCHCLOSE command but the financial processor associated with the account controls batch processing.
RC_CASSETTE_BATCH_CURRENCY	1011	The currency for all transactions in a batch must be the same.
RC_CASSETTE_BATCH_AMOUNTEXP10	1012	The amount exponent for all transactions in a batch must be the same.
RC_CASSETTE_BRAND	1014	Response refers to the brand parameter (specified in protocol data).
RC_CASSETTE_PAN	1015	Response refers to the PAN parameter (specified in protocol data).
RC_CASSETTE_EXPIRY	1016	Response refers to the expiry parameter (specified in protocol data).
RC_CASSETTE_DEPOSIT_AMOUNT_NOT_ZERO	1017	This account only allows complete deposit reversals (that is, the amount must be zero).
RC_CASSETTE_COMMUNICATION_ERROR	1018	A communication error occurred between the cassette and an entity with which it communicates.
RC_CASSETTE_INTERMEDIATE_RESPONSE_NULL	1019	The cassette received a unexpected NULL response from an entity with which it communicates.
RC_CASSETTE_INTERMEDIATE_RESPONSE_UNEXPECTED	1020	The cassette received a unexpected response from an entity with which it communicates.
RC_CASSETTE_BATCH_ERROR	1021	A batch-related error occurred.
RC_CASSETTE_BATCH_BALANCE_ERROR	1022	The totals for this batch calculated by the WebSphere Commerce Payments and the financial institution did not match.
RC_CASSETTE_APPROVE_NO_DEPOSIT	1040	While processing an APPROVE with automatic deposit, the cassette successfully completed the approval, but could not successfully complete the deposit.
RC_CASSETTE_DECLINED	1041	The financial institution declined the request for an unknown reason.

Table 29. Generic Secondary Return Codes (SRCs) (continued)

RC_CASSETTE_DECLINED_EXPIRY	1042	The financial institution declined the request due to the expiry value.
RC_CASSETTE_DECLINED_INSTRUMENT	1043	The financial institution declined the request due to a problem with the purchase instrument (the credit card, check or whatever instrument is used by this cassette's payment protocol).
RC_CASSETTE_AVSDATA	1051	Response refers to the group of AVS parameters (specified in protocol data).
RC_CASSETTE_AVS_COUNTRYCODE	1052	Response refers to the AVS country code parameter (specified in protocol data).
RC_CASSETTE_AVS_STREETADDRESS	1053	Response refers to the AVS street address parameter (specified in protocol data).
RC_CASSETTE_AVS_CITY	1054	Response refers to the AVS city parameter (specified in protocol data).
RC_CASSETTE_AVS_STATEPROVINCE	1055	Response refers to the AVS state/province parameter (specified in protocol data).
RC_CASSETTE_AVS_POSTALCODE	1056	Response refers to the AVS postal code parameter (specified in protocol data).
RC_CASSETTE_AVS_LOCATIONID	1057	Response refers to the AVS location id parameter (specified in protocol data).
RC_CASSETTE_CARDHOLDERNAME	1058	Response refers to the cardholder name parameter (specified in protocol data).
RC_CASSETTE_MAXBATCHSIZE	1059	Response refers to the maximum batch size parameter (specified in protocol data).
RC_CASSETTE_CURRENCY	1060	Response refers to the currency parameter (specified in protocol data).
RC_CASSETTE_HUMAN_INTERVENTION_REQUIRED	1061	The operation failed completely or partially. Human intervention is required to resolve the failure.
RC_CASSETTE_DECLINED_APPROVAL_EXPIRED	1062	The approval for the payment has expired. You must obtain a new approval for the payment amount before you can successfully deposit. If the cassette supports ApproveReversal, then use it to obtain the new approval for the existing payment. Otherwise, use Approve to create a new approved payment which you can subsequently deposit.
RC_CASSETTE_AMOUNT_WOULD_EXCEED_ORDER_AMOUNT	1063	Approval of the specified amount would cause the cumulative amount of all payments exceed the original order amount.
RC_CASSETTE_VERSION	1064	Cassette version specified in the database table exceeds the maximum length.

Table 29. Generic Secondary Return Codes (SRCs) (continued)

RC_CASSETTE_CARDVERIFYCODE	1065	Response refers to the specified card verification code.
RC_CASSETTE_AUTHCODE	1066	Response refers to the specified authorization code.
RC_CASSETTE_DECLINECODE	1067	Response refers to the specified decline code.
RC_REALM_INIT_ERROR	1068	The defined realm could not be initialized.
RC_REALM_OPERATION_ERROR	1069	An error occurred while using the defined realm.
RC_CASSETTE_SHIPPINGDATA	1071	Response refers to the group of shipping address parameters (specified in protocol data).
RC_CASSETTE_SHIP_COUNTRYCODE	1072	Response refers to the shipping country code parameter (specified in protocol data).
RC_CASSETTE_SHIP_STREETADDRESS	1073	Response refers to the shipping street address parameter (specified in protocol data).
RC_CASSETTE_SHIP_CITY	1074	Response refers to the shipping city parameter (specified in protocol data).
RC_CASSETTE_SHIP_STATEPROVINCE	1075	Response refers to the shipping state/province parameter (specified in protocol data).
RC_CASSETTE_SHIP_POSTALCODE	1076	Response refers to the shipping postal code parameter (specified in protocol data).
RC_CASSETTE_BILLINGDATA	1081	Response refers to the group of billing address parameters (specified in protocol data).
RC_CASSETTE_BILL_COUNTRYCODE	1082	Response refers to the billing country code parameter (specified in protocol data).
RC_CASSETTE_BILL_STREETADDRESS	1083	Response refers to the billing street address parameter (specified in protocol data).
RC_CASSETTE_BILL_CITY	1084	Response refers to the billing city parameter (specified in protocol data).
RC_CASSETTE_BILL_STATEPROVINCE	1085	Response refers to the billing state/province parameter (specified in protocol data).
RC_CASSETTE_BILL_POSTALCODE	1086	Response refers to the billing postal code parameter (specified in protocol data).
RC_ACCEPTPAYMENTAUTOAPPROVE	1087	Response refers to the approve flag on the merchant account on AcceptPayment.
RC_ACCEPTPAYMENTAUTODEPOSIT	1088	Response refers to the deposit flag on the merchant account on AcceptPayment.

Table 29. Generic Secondary Return Codes (SRCs) (continued)

RC_RECEIVEPAYMENTAUTOAPPROVE	1089	Response refers to the approve flag on the merchant account on ReceivePayment.
RC_RECEIVEPAYMENTAUTODEPOSIT	1090	Response refers to the deposit flag on the merchant account on ReceivePayment.
RC_CASSETTE_COUNTRYCODE	1092	Response refers to the country code parameter (specified in protocol data).
RC_CASSETTE_STREETADDRESS	1093	Response refers to the street address parameter (specified in protocol data).
RC_CASSETTE_CITY	1094	Response refers to the city parameter (specified in protocol data).
RC_CASSETTE_STATEPROVINCE	1095	Response refers to the state or province parameter (specified in protocol data).
RC_CASSETTE_POSTALCODE	1096	Response refers to the postal (zip) code parameter (specified in protocol data).
RC_CASSETTE_AVSCODE	1097	Response refers to the AVS code parameter (specified in protocol data).
RC_CASSETTE_AUTHCODE_AND_DECLINEREASON	1098	Conflicting protocol data was specified with this API command.
RC_CASSETTE_BATCHCLOSETIME	1099	Response refers to the batch close time parameter (specified in protocol data).
RC_CASSETTE_METHOD	1100	Response refers to the payment method parameter (specified in protocol data).
RC_CASSETTE_FIBATCHID	1101	Response refers to the financial institution batch identification parameter (specified in protocol data).
RC_CASSETTE_AUXILIARY1	1102	Response refers to the first auxiliary text parameter (specified in protocol data).
RC_CASSETTE_AUXILIARY2	1103	Response refers to the second auxiliary text parameter (specified in protocol data).
RC_CASSETTE_DECLINEREASON	1104	Response refers to the specified authorization reason.
RC_CASSETTE_BUYERNAME	1105	Response refers to the Buyer Name.
RC_CASSETTE_STREETADDRESS2	1106	Response refers to the Street Address, Line 2.
RC_CASSETTE_PHONENUMBER	1107	Response refers to the phone number.
RC_CASSETTE_EMAILADDRESS	1108	Response refers to the email address.
RC_CASSETTE_CHECKROUTINGNUMBER	1109	Response refers to the check routing number.
RC_CASSETTE_CHECKINGACCOUNTNUMBER	1110	Response refers to the checking account number.

Appendix B. ISO currency codes

Following is a list of ISO 4217 currency codes. Use these values with the CURRENCY parameter.

Country/Region	Code Alpha	Code Numeric	Exponent Conversions	Currency
Afganistan	AFA	004	-2	Afghanistan Afghani
Albania	ALL	008	-2	Albanian Lek
Algeria	DZD	012	-2	Algerian Dinar
American Samoa	USD	840	-2	US Dollar
Andorra	ESP, FRF, ADP	724, 250, 020	0, -2, 0	Spanish Peseta, French Franc, Andorran Peseta
Angola	AOA	973	-2	Kwanza
Anguilla	XCD	951	-2	East Caribbean Dollar
Antigua and Barbuda	XCD	951	-2	East Caribbean Dollar
Argentina	ARS	032	-2	Argentine Peso
Armenia	AMD	051	-2	Armenian Dram
Aruba	AWG	533	-2	Aruban Guilder
Australia	AUD	036	-2	Australian Dollar
Austria	ATS	040	-2	Austrian Schilling
Azerbaijan	AZM	031	-2	Azerbaijanian Manat
Bahamas	BSD	044	-2	Bahamian Dollar
Bahrain	BHD	048	-3	Bahraini Dinar
Bangladesh	BDT	050	-2	Bangladeshi Taka
Barbados	BBD	052	-2	Barbados Dollar
Belarus	BYB, RYR	112, 974	0	Belarussian Ruble, Belarussian Ruble
Belgium	BEF	056	0	Belgian Franc
Belize	BZD	084	-2	Belize Dollar

Benin	XOF	952	0	CFA Franc (BCEAO)
Bermuda	BMD	060	-2	Bermuda Dollar
Bhutan	INR, BTN	356, 064	-2, -2	Indian Rupee, Ngultrum
Bolivia	BOB, BOV	068, 984	-2, -2	Boliviano, Mvdol
Bosnia & Herzegovina	BAM	977	-2	Convertible Marks
Botswana	BWP	072	-2	Pula
Bouvet Island	NOK	578	-2	Norwegian Krone
Brazil	BRL	986	-2	Brazil Real
British Indian Ocean Territory	USD	840	-2	US Dollar
Brunei Darrusslam	BND	096	-2	Brunei Dollar
Bulgaria	BGL, BGN	100, 975	-2, -2	Lev, Bulgarian Lev
Burkina Faso	XOF	952	0	CFA Franc BCEAO
Burundi	BIF	108	0	Burundi Franc
Cambodia	KHR	116	-2	Cambodian Riel
Cameroon	XAF	950	0	CFA Franc (BEAC)
Canada	CAD	124	-2	Canadian Dollar
Cape Verde	CVE	132	-2	Cape Verde Escudo
Cayman Islands	KYD	136	-2	Cayman Islands Dollar
Central African Republic	XAF	950	0	CFA Franc (BEAC)
Chad	XAF	950	0	CFA Franc (BEAC)
Chile	CLP, CLF	152, 990	0, 0	Chilean Peso, Unidates de fomento
China	CNY	156	-2	Yuan Renminbi
China (Hong Kong S.A.R.)	HKD	344	-2	Hong Kong Dollar
China (Mancau S.A.R.)	MOP	446	-2	Pataca
Christmas Island	AUD	036	-2	Australian Dollar
Cocos (Keeling) Islands	AUD	036	-2	Australian Dollar

Colombia	COP	170	-2	Colombian Peso
Comoros	KMF	174	0	Comoro Franc
Congo	XAF	950	0	CFA Franc (BEAC)
Congo, Democratic Republic of	CDF	976	-2	Franc Congolais
Cook Islands	NZD	554	-2	New Zealand Dollar
Costa Rica	CRC	188	-2	Costa Rican Colon
Côte D'Ivoire	XOF	952	0	CFA Franc (BCEAO)
Croatia	HRK	191	-2	Croatian Kuna
Cuba	CUP	192	-2	Cuban Peso
Cyprus	CYP	196	-2	Cyprus Pound
Czech Republic	CZK	203	-2	Czech Koruna
Denmark	DKK	208	-2	Danish Krone
Djibouti	DJF	262	0	Djibouti Franc
Dominica	XCD	951	-2	East Caribbean Dollar
Dominican Republic	DOP	214	-2	Dominican Peso
East Timor	TPE, IDE	626, 360	0, -2	Timor Escudo, Rupiah
Ecuador	ECS, ECV	218, 983	-2, -2	Sucre, Unidad de Valor Constante (UVC)
Egypt	EGP	818	-2	Egyptian Pound
El Salvador	SVC	222	-2	El Salvador Colon
Equatorial Guinea	XAF	950	0	CFA Franc (BEAC)
Eritrea	ERN	232	-2	Nafka
Estonia	EEK	233	-2	Kroon
Ethiopia	ETB	230	-2	Ethiopian Birr
Faroe Islands	DKK	208	-2	Danish Krone
European Union (ECU)	XEU	954	-2	euro
European Union (Euro)	EUR	978	-2	European Currency Unit
Falkland Islands	FKP	238	-2	Falkland Islands Pound
Fiji	FJD	242	-2	Fiji Dollar
Finland	FIM	246	-2	Finnish Markka

France	FRF	250	-2	French Franc
French Guiana	FRF	250	-2	French Franc
French Polynesia	XPF	953	0	CFP Franc
French Southern Territories	XPF	953	0	CFP Franc
Gabon	XAF	950	0	CFA Franc (BEAC)
Gambia	GMD	270	-2	Dalasi
Georgia	GEL	981	-2	Lari
Germany	DEM	276	-2	Deutsche Mark
Ghana	GHC	288	-2	Ghana Cedi
Gibraltar	GIP	292	-2	Gibraltar Pound
Greece	GRD	300	0	Drachma
Greenland	DKK	208	-2	Danish Krone
Granada	XCD	951	-2	East Caribbean Dollar
Guadaloupe	FRF	250	-2	French Franc
Guam	USD	840	-2	US Dollar
Guatemala	GTQ	320	-2	Guatemalan Quetzal
Guinea	GNF	324	0	Guinea Franc
Guinea-Bissau	GWP, XOF	624, 952	-2, 0	Guinea-Bissau Peso, CFA Franc (BCEAO)
Guyana	GYD	328	-2	Guyana Dollar
Haiti	HTG, USD	332, 840	-2, -2	Haiti Gourde, US Dollar
Heard Island and McDonald Islands	AUD	036	-2	Australian Dollar
Holy See (Vatican City State)	ITL	380	0	Italian Lira
Honduras	HNL	340	-2	Honduran Lempira
Hungary	HUF	348	-2	Forint
Iceland	ISK	352	-2	Iceland Krona
India	INR	356	-2	Indian Rupee
Indonesia	IDR	360	-2	Indonesian Rupiah
International Monetary Fund	XDR	960	N.A.	SDR
Iran	IRR	364	-2	Iranian Rial
Iraq	IQD	368	-3	Iraqi Dinar
Ireland	IEP	372	-2	Irish Pound

Israel	ILS	376	-2	New Israeli Sheqel
Italy	ITL	380	0	Italian Lira
Jamaica	JMD	388	-2	Jamaican Dollar
Japan	JPY	392	0	Yen
Jordan	JOD	400	-3	Jordanian Dinar
Kazakhstan	KZT	398	-2	Kazakhstan Tenge
Kenya	KES	404	-2	Kenyan Shilling
Kiribati	AUD	036	-2	Australian Dollar
Korea, Democratic People's Republic of	KPW	408	-2	North Korean Won
Korea, Republic of	KRW	410	0	South Korean Won
Kuwait	KWD	414	-3	Kuwaiti Dinar
Kyrgyzstan	KGS	417	-2	Kyrgyzstan Som
Lao People's Democratic Republic	LAK	418	-2	Laos Kip
Latvia	LVL	428	-2	Latvian Lats
Lebanon	LBP	422	-2	Lebanese Pound
Lesotho	ZAR, LSL	710, 426	-2, -2	Rand, Loti
Liberia	LRD	430	-2	Liberian Dollar
Libyan Arab Jamahirya	LYD	434	-3	Libyan Dinar
Liechtenstein	CHF	756	-2	Swiss Franc
Lithuania	LTL	440	-2	Lithuanian Litas
Luxembourg	LUF	442	0	Luxembourg Franc
Macedonia (Former Yug. Rep.)	MKD	807	-2	Macedonian Denar
Madagascar	MGF	450	0	Malagasy Franc
Malawi	MWK	454	-2	Kwacha
Malaysia	MYR	458	-2	Malaysian Ringgit
Maldives	MVR	462	-2	Maldives Rufiyaa
Mali	XOF	952	0	CFA Franc BCEAO
Malta	MTL	470	-2	Maltese Lira
Marshall Islands	USD	840	-2	US Dollar

Martinique	FRF	250	-2	French Franc
Mauritania	MRO	478	-2	Mauritanian Ouguiya
Mauritius	MUR	480	-2	Mauritius Rupee
Mexico	MXN, MXV	484, 979	-2, -2	Mexican Peso, Mexican Unidad de Inversion (UDI)
Micronesia	USD	840	-2	US Dollar
Moldova, Republic of	MDL	498	-2	Moldovan Leu
Monaco	FRF	250	-2	French Franc
Mongolia	MNT	496	-2	Mongolian Tugrik
Montserrat	XCD	951	-2	East Caribbean Dollar
Morocco	MAD	504	-2	Moroccan Dirham
Mozambique	MZM	508	-2	Mozambique Metical
Myanmar	MMK	104	-2	Myanmar Kyat
Namibia	ZAR, NAD	710, 516	-2, -2	Rand, Namibia Dollar
Nauru	AUD	036	-2	Australian Dollar
Nepal	NPR	524	-2	Nepalese Rupee
Netherlands Antilles	ANG	532	-2	Netherlands Antillian Guilder
Netherlands	NLG	528	-2	Netherlands Gulder
New Caledonia	XPF	953	0	CFP Franc
New Zealand	NZD	554	-2	New Zealand Dollar
Nicaragua	NIO	558	-2	Nicaraguan Cordoba Oro
Niger	XOF	952	0	CFA Franc BCEAO
Nigeria	NGN	566	-2	Nigerian Naira
Niue	NZD	554	-2	New Zealand Dollar
Norfolk Island	AUD	036	-2	Australian Dollar
Northern Mariana Islands	USD	840	-2	US Dollar
Norway	NOK	578	-2	Norwegian Krone

Oman	OMR	512	-3	Rial Omani
Pakistan	PKR	586	-2	Pakistan Rupee
Palau	USD	840	-2	US Dollar
Panama	PAB, USD	590, 840	-2, -2	Balboa, US Dollar
Papua New Guinea	PGK	598	-2	Papua New Guinea Kina
Paraguay	PYG	600	0	Paraguay Guarani
Peru	PEN	604	-2	Peru Nuevo Sol
Philippines	PHP	608	-2	Philippine Peso
Pitcairn	NZD	554	-2	New Zealand Dollar
Poland	PLN	985	-2	Poland Zloty
Portugal	PTE	620	0	Portuguese Escudo
Puerto Rico	USD	840	-2	US Dollar
Qatar	QAR	634	-2	Qatari Rial
Reunion	FRF	250	-2	French Franc
Romania	ROL	642	-2	Romanian Leu
Russian Federation	RUR, RUB	810, 643	-2, -2	Russian Ruble, Russian Ruble
Rwanda	RWF	646	0	Rwanda Franc
Saint Kitts and Nevis	XCD	951	-2	East Caribbean Dollar
Saint Lucia	FRF	951	-2	East Caribbean Dollar
Saint Pierre and Miquelon	XCD	250	-2	French Franc
Saint Vincent and the Grenadines	XCD	951	-2	East Caribbean Dollar
Saint Helena	SHP	654	-2	St. Helena Pound
Samoa	WST	882	-2	Tala
San Marino	ITL	380	0	Italian Lira
Sao Tome and Principe	STD	678	-2	Sao Tome and Principe Dobra
Saudi Arabia	SAR	682	-2	Saudi Riyal
Senegal	XOF	952	0	CFA Franc BCEAO
Seychelles	SCR	690	-2	Seychelles Rupee

Sierra Leone	SLL	694	-2	Sierra Leone Leone
Singapore	SGD	702	-2	Singapore Dollar
Slovakia	SKK	703	-2	Slovak Koruna
Slovenia	SIT	705	-2	Slovenia Tolar
Solomon Island	SBD	090	-2	Solomon Islands Dollar
Somalia	SOS	706	-2	Somalia Shilling
South Africa	ZAR	710	-2	South African Rand
Spain	ESP	724	0	Spanish Peseta
Sri Lanka	LKR	144	-2	Sri Lanka Rupee
Sudan	SDP	736	-2	Sudanese Dinar
Suriname	SRG	740	-2	Suriname Guilder
Svalbard and Jan Mayen	NOK	578	-2	Norwegian Krone
Swaziland	SZL	748	-2	Swaziland Lilangeni
Sweden	SEK	752	-2	Swedish Krona
Switzerland	CHF	756	-2	Swiss Franc
Syrian Arab Republic	SYP	760	-2	Syrian Pound
Taiwan	TWD	901	-2	New Taiwan Dollar
Tajikistan	TJR	762	0	Tajik Ruble
Tanzania, United Republic of	TZS	834	-2	Tanzanian Shilling
Thailand	THB	764	-2	Thai Baht
Togo	XOF	952	0	CFA Franc BCEAO
Tokelau	NZD	554	-2	New Zealand Dollar
Tonga	TOP	776	-2	Tonga Pa'anga
Trinidad and Tobago	TTD	780	-2	Trinidad and Tobago Dollar
Tunisia	TND	788	-3	Tunisian Dinar
Turkey	TRL	792	0	Turkish Lira
Turkmenistan	TMM	795	-2	Manat
Turks and Caicos Islands	USD	840	-2	US Dollar
Tuvalu	AUD	036	-2	AUD

Uganda	UGX	800	2	Ugandan Shilling
Ukraine	UAH	980	-2	Hryvnia
United Arab Emirates	AED	784	-2	UAE Dirham
United Kingdom	GBP	826	-2	Pound Sterling
United States of America	USD, USS, USN	840, 998, 997	-2, -2, -2	US Dollar, (Same day) (Next day)
United States Minor Outlying Islands	USD	840	-2	US Dollar
Uruguay	UYU	858	-2	Peso Uruguayo
Uzbekistan	UZS	860	-2	Uzbekistan Sum
Vanuatu	VUV	548	0	Vanuatu Vatu
Venezuela	VEB	862	-2	Venezuela Bolivar
Viet Nam	VND	704	-2	Viet Nam Dong
Virgin Islands (British)	USD	840	-2	US Dollar
Virgin Islands (US)	USD	840	-2	US Dollar
Wallis and Futuna	XPF	953	0	CFP Franc
Western Sahara	MAD	504	-2	Moroccan Dirham
Yemen	YER	886	-2	Yemeni Rial
Yugoslavia	YUN	891	-2	Yugoslavian Dinar
Zaire	ZRN	180	-2	Unknown
Zambia	ZMK	894	-2	Zambia Kwacha
Zimbabwe	ZWD	716	-2	Zimbabwe Dollar

Appendix C. Obtaining requests for comments

Requests for comments (RFCs) are documents that present new protocols and establish standards for the Internet protocol suite. Hardcopies of all RFCs are available from the Network Information Center (NIC), either individually or on a subscription basis. You can obtain these documents from:

Government Systems, Inc.
Attn: Network Information Center
14200 Park Meadow Drive
Suite 200
Chantilly, VA 22021

You can access RFCs from this URL:

http://www.cis.ohio-state.edu/hypertext/information/rfc.html

Appendix D. Notices

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

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

IBM Director of Licensing
IBM Corporation
500 Columbus Avenue
Thornwood, NY 10594
U.S.A.

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

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

The following paragraph does not apply to the United Kingdom or any other country where such provisions are inconsistent with local law:
INTERNATIONAL BUSINESS MACHINES CORPORATION PROVIDES THIS PUBLICATION "AS IS" WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESS OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF NON-INFRINGEMENT, MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE. Some states do not allow disclaimer of express or implied warranties in certain transactions, therefore, this statement may not apply to you.

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

Any references in this publication to non-IBM Web sites are provided for convenience only and do not in any manner serve as an endorsement of those Web sites. The materials at those Web sites are not part of the materials for this IBM product and use of those Web sites is at your own risk.

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

IBM Corporation

Department TL3B/Building 062
PO Box 12195
3039 Cornwallis Road
Research Triangle Park, NC 27709-2195

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

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

Trademarks

The following terms are trademarks of the IBM Corporation in the United States or other countries or both:

- AIX
- DB2
- IBM
- IBM Consumer Wallet
- IBM Payment Gateway
- IBM Payment Registry
- IBM Payment Server
- iSeries
- RS/6000

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

Lotus and Lotus Domino Go Webserver are trademarks of Lotus Development Corporation in the United States and/or other countries.

Microsoft, Windows NT, and the Windows logo are trademarks or registered trademarks of Microsoft Corporation in the United States and/or other countries.

SET Secure Electronic Transaction, Secure Electronic Transaction, SET, and the SET Secure Electronic Transaction design mark are trademarks and service marks owned by SET Secure Electronic Transaction LLC.

UNIX is a registered trademark in the United States and/or other countries licensed exclusively through X/Open Company Limited.

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

Glossary

This dictionary defines technical terms used in the documentation for Payment Suite products. It includes IBM product terminology and may include selected terms and definitions from:

- The *American National Standard Dictionary for Information Systems* , ANSI X3.172-1990, copyright 1990 by the American National Standards Institute (ANSI). Copies may be purchased from the American National Standards Institute, 11 West 42nd Street, New York, New York 10036. Definitions are identified by the symbol (A) after the definition.
- The ANSI/EIA Standard—440-A, *Fiber Optic Terminology*. Copies may be purchased from the Electronic Industries Association, 2001 Pennsylvania Avenue, N.W., Washington, DC 20006. Definitions are identified by the symbol (E) after the definition.
- The *Information Technology Vocabulary* developed by Subcommittee 1, Joint Technical Committee 1, of the International Organization for Standardization and the International Electrotechnical Commission (ISO/IEC JTC1/SC1). Definitions of published parts of this vocabulary are identified by the symbol (I) after the definition; definitions taken from draft international standards, committee drafts, and working papers being developed by ISO/IEC JTC1/SC1 are identified by the symbol (T) after the definition, indicating that final agreement has not yet been reached among the participating National Bodies of SC1.
- The *IBM Dictionary of Computing*, New York: McGraw-Hill, 1994.
- Internet Request for Comments: 1208, *Glossary of Networking Terms*
- Internet Request for Comments: 1392, *Internet Users' Glossary*
- The *Object-Oriented Interface Design: IBM Common User Access Guidelines* , Carmel, Indiana: Que, 1992.

The most current *IBM Dictionary of Computing* is available on the World Wide Web at http:\\www.ibm.com/networking/nsq/nsqmain.htm.

The following cross-references are used in this dictionary:

Contrast with:
This refers the reader to a term that has an opposed or substantively different meaning.

See: This refers the reader to (a) a related term, (b) a term that is the expanded form of an abbreviation or acronym, or (c) a synonym or more preferred term.

Obsolete term for:
This indicates that the term should not be used and refers the reader to the preferred term.

A

access control. In computer security, the process of ensuring that the resources of a computer system can be accessed only by authorized users in authorized ways.

account. An account is a relationship between the merchant and the financial institution which processes transactions for that merchant. There can be multiple accounts for each payment cassette.

account name. The name you assign to the account. Its only function is to provide display information in the user interface.

acquirer. In e-commerce, the financial institution (or an agent of the financial institution) that receives from the merchant the financial data relating to a transaction and initiates that data into an interchange system.

Address Verification Service (AVS). Within IBM e-commerce, a credit and debit card scheme used by merchants to authenticate the cardholder. The merchant requests the cardholder's address and uses AVS to confirm that the cardholder is who he says he is.

ADSM. See ADSTAR Distributed Storage Manager.

ADSTAR Distributed Storage Manager (ADSM). An IBM licensed program that provides storage management and data access services in a multi-vendor, distributed computing environment.

applet. An application program, written in the Java programming language, that can be retrieved from a Web server and executed by a Web browser. A reference to an applet appears in the markup for a Web page, in the same way that a reference to a graphics file appears; a browser retrieves an applet in the same way that it retrieves a graphics file. For security reasons, an applet's access rights are limited in two ways: the applet cannot access the file system of the

client upon which it is executing, and the applet's communication across the network is limited to the server from which it was downloaded. Contrast with servlet.

approve. Within IBM e-commerce, a WebSphere Commerce Payments verb. A merchant issues this verb to create a Payment object. For cassettes that implement credit card protocols, this verb will likely map to authorization (see authorize). Other cassettes may implement the approval process differently. For IBM WebSphere Commerce Payments Cassette for SET and Cassette for CyberCash, the **approve** verb results in the creation of a Payment object and authorization to ensure that funds are available to cover payment.

approve all. Selects all orders displayed for approval.

approved amount. The amount of the order approved for payment.

approve selected. Selects the orders that you want to create a payment in the approved state for. You must perform a manual deposit on this payment to move it from approved state to deposit state.

asymmetric. In computer security, pertaining to the use of different keys for encryption and decryption.

authentication. (1) In SETCo., the process that seeks to validate identity or to prove the integrity of the information. Authentication in public key systems uses digital signatures. (2) In computer security, verification that a message has not been altered or corrupted. (3) In computer security, a process used to verify the user of an information system or protected resources.

authorization. (1) In SETCo., the process by which a properly appointed person or persons grants permission to perform some action on behalf of an organization. This process assesses transaction risk, confirms that a given transaction does not raise the account holder debt above the account credit limit, and reserves the specified amount of credit. (When a merchant obtains authorization, payment for the authorized amount is guaranteed provided that the merchant followed the rules associated with the authorization process.) (2) In computer security, the right granted to a user to communicate with or make use of a computer system. (T) (3) An access right. (4) The process of granting a user either complete or restricted access to an object, resource, or function.

authorization reversal. In SETCo., a transaction sent when a previous authorization needs to be canceled (that is, a full reversal performed) or decreased (that is, a partial reversal performed). A full reversal will be used when the transaction cannot be completed, such as when the cardholder cancels the order or the merchant discovers that goods are no longer available, as when discontinued. A partial reversal will be used when the authorization was for the entire order and some of the goods cannot be shipped, resulting in a split shipment.

authorize. In the credit card world, a merchant is guaranteed that cardholder funds are available to cover a transaction by first *authorizing* the transaction. The cardholder's issuer (that is, the bank that issued the card) guarantees payment.

B

balance. Within IBM e-commerce, an attribute of a WebSphere Commerce Payments Batch object. Indicates whether the merchant and financial institution agreed on the contents of the batch when it was closed. See 172 for more details.

balanced. Within IBM e-commerce, an attribute of a WebSphere Commerce Payments Batch object. The batch has been successfully balanced. All totals agree.

balance status. Within IBM e-commerce, an attribute of a WebSphere Commerce Payments Batch object. The balance status of a batch can be balanced or out of balance.

baseline. In SETCo., a baseline product is the specific product within an operating system family that is run against the SET Tests. A vendor must designate a distinct baseline product for each unrelated operating system family. Refer to the *SET Testing Policies and Procedures* for a complete explanation.

batch. (1) In the credit card world , a batch is a collection of fund transfer requests that are all done at the same time (that is, *in a batch*). Individual fund transfers are not performed for each individual payment, but, rather, a group of transfers is processed so that both the merchant and the financial institution can agree on the funds that are to be transferred. Before a batch is *closed* (that is, the funds are exchanged) there is usually some type of reconciliation process so that both sides agree on the amounts. A group of records or data processing jobs brought together for processing or transmission. (2) Within IBM e-commerce, one of the fundamental WebSphere Commerce Payments objects is the Batch. A Batch is an object with which Payment and Credit objects are associated. Transfer of funds is to occur when the batch is closed. (3) A group of records or data processing jobs brought together for processing or transmission.

batch close date. One of two numeric search parameters that defines the chronological start of your search. Specify a date that precedes the batch close date for the batch you want to search.

batch number. The number that identifies the batch. WebSphere Commerce Payments assigns a number to the batch when the payment is deposited.

batch open date. One of two numeric search parameters that defines the chronological start of your search. Specify a date that precedes the batch open date for the batch you want to search.

batch number. The number that identifies the batch. The number WebSphere Commerce Payments assigns to the batch when the payment is deposited.

batch search. Search for a single batch or group of batches, based on a defined list of characteristics.

BCD. See binary-coded decimal notation.

big endian. A format for storage or transmission of binary data in which the most significant bit (or byte) is placed first. Contrast with little endian.

binary-coded decimal (BCD) notation. A binary-coded notation in which each of the decimal digits is represented by a binary numeral; for example, in binary-coded decimal notation that uses the weights 8, 4, 2, 1, the number "twenty-three" is represented by 0010 0011 (compare its representation 10111 in the pure binary numeration system). (I) (A)

bitmapped message. A variable-length transaction in which each bit in an array of bits indicates the presence or absence of a data field within the transaction.

brand. Within IBM e-commerce, the Cassette object for all of the WebSphere Commerce Payments cassettes (for example, Cassette for SET and Cassette for CyberCash). Each financial transaction for a WebSphere Commerce Payments cassette is associated with a particular brand (for example, MasterCard or VISA). Each account with a financial institution can be configured to support one or more brands.

browser. See Web browser.

browser plug-in. See Web browser plug-in.

C

CA. See certificate authority.

capture. (1) In SETCo., a transaction sent after the merchant has shipped the goods. This transaction will trigger the movement of funds from the Issuer to the Acquirer and then to the merchant account. (2) In the credit card world, payment is actually made when the funds are *captured*. All payments must be authorized *and* captured (although this work can be done at the same time). Note that all payments are associated with a batch (see "void payment" on page 173) and that the actual capture occurs when the associated batch is closed.

capture reversal. In SETCo., a transaction sent when the information in a previous capture message was incorrect or should never have been sent (such as when the goods were not actually shipped). If the capture reversal is the result of incorrect information, it will be followed by a new capture message with the correct information.

cardholder. In e-commerce, a person who has a valid payment card account and uses software that supports e-commerce.

cardholder application. In SETCo., a cardholder application, sometimes called a wallet, that is run by an online consumer that enables secure payment card transactions over a network. SET Cardholder applications must generate SET protocol messages that can be accepted by SET Merchant, Payment Gateway, and Certificate Authority components.

cascading. In high-availability cluster multiprocessing (HACMP), pertaining to a cluster configuration in which the cluster node with the highest priority for a particular resource acquires the resource if the primary node fails but relinquishes the resource to the primary node upon reintegration of the primary node into the cluster. Contrast with concurrent and rotating.

cassette. (1) In e-commerce, a software component consisting of a collection of Java classes and interfaces that can be easily installed into other software components involved in e-commerce to extend the function of these components. (2) In IBM e-commerce, a WebSphere Commerce Payments concept. The WebSphere Commerce Payments provides a framework that can support many different forms of payment. WebSphere Commerce Payments cassettes are written by IBM or third-party vendors to support different payment protocols (such as, SET and CyberCash) within the WebSphere Commerce Payments Framework. Thus, WebSphere Commerce Payments is an extensible product that can support additional protocols.

cast. In programming languages, an operator that converts the value of its operand to a specified type.

CERN. Conseil Européen pour la Recherche Nucléaire (European Laboratory for Particle Physics). Located in Geneva, Switzerland, CERN initiated the World Wide Web and was the first organization to create a Web server. The CERN Web server is the basis for many commercially available servers.

certificate. (1) In e-commerce, a digital document that binds a public key to the identity of the certificate owner, thereby enabling the certificate owner to be authenticated. A certificate is issued by a certificate authority (CA). (2) In SETCo., a certificate that has been digitally signed by a trusted authority (usually the cardholder financial institution) to identify the user of the public key. SET defines the following certificate types:

- signature
- key encipherment
- certificate signature
- CRL signature

certificate authority (CA). (1) In e-commerce, an organization that issues certificates. The CA

authenticates the certificate owner's identity and the services that the owner is authorized to use, issues new certificates, renews existing certificates, and revokes certificates belonging to users who are no longer authorized to use them. See issuer. (2) In SETCo., certificate authority refers to both the component and to the entity issuing and verifying digital certificates. The component is a product run by a Certificate Authority entity that is authorized to issue and verify digital certificates as requested by Cardholder Wallet components, Merchant Server components, and/or Payment Gateway components over public and private networks.

certificate chain. (1) In SETCo., a hierarchy of digital certificates. The certificate at the top of the hierarchy is called the ″root certificate.″ (2) In SETCo., an ordered grouping of digital certificates, including the root certificate, that are used to validate a specific certificate.

certificate renewal. In SETCo., the process by which a new certificate is created for an existing public key.

certificate revocation. In SETCo., the process of revoking an otherwise valid certificate by the entity that issued the certificate.

certificate revocation list. In SETCo., a list of certificate serial numbers previously issued by a certificate authority that indicate the certificates that are invalid prior to normal expiration due to compromise, disaffiliation, or some other unusual circumstance.

certification. In SETCo., the process of ascertaining that a set of requirements or criteria has been fulfilled and attesting to that fact to others, usually with some written instrument. A system that has been inspected and evaluated as fully compliant with the SET protocol by duly authorized parties and process would be said to have been certified.

certification authority. See certificate authority.

certified. In SETCo., the process of ascertaining that a set of requirements or criteria has been fulfilled and attesting to that fact to others, usually with some written instrument. A system that has been inspected and evaluated as fully compliant with the SET protocol by duly authorized parties and process would be said to have been certified.

CGI. See Common Gateway Interface.

CGI program. A computer program that runs on a Web server and uses the Common Gateway Interface (CGI) to perform tasks that are not usually done by a Web server (for example, database access and form processing). A CGI script is a CGI program that is written in a scripting language such as Perl.

CGI script. See CGI program.

clerk. (1) In IBM e-commerce, this is a WebSphere Commerce Payments concept. The WebSphere Commerce Payments has four different access rights. A clerk is defined on a per-merchant basis and has the lowest level of access. (2) A clerk is a low-level employee.

client. A computer system or process that requests a service of another computer system or process that is typically referred to as a server. Multiple clients may share access to a common server.

closed. An order moves into closed state when its associated payment, or payments, moves from deposited state into closed state (that is, when the batch associated with the payment closes). When an order is in closed state, the financial transaction is complete; monies are deposited, and the order cannot be modified. No commands are permitted for orders in this state.

cluster. In high-availability cluster multiprocessing (HACMP), a set of independent systems (called nodes) that are organized into a network for the purpose of sharing resources and communicating with each other.

cluster node. In high-availability cluster multiprocessing (HACMP), an RS/6000 system that participates in a cluster.

commerce service provider (CSP). An Internet service provider that hosts merchant shopping sites and processes payments for the merchants.

Common Gateway Interface (CGI). A standard for the exchange of information between a Web server and computer programs that are external to it. The external programs can be written in any programming language that is supported by the operating system on which the Web server is running. See CGI program.

concurrent. In high-availability cluster multiprocessing (HACMP), pertaining to a cluster configuration in which all cluster nodes use a resource simultaneously. A cluster node can fail and reintegrate into the cluster without affecting other cluster nodes or the resource. Contrast with cascading and rotating.

confidentiality. In SETCo., the protection of sensitive and personal information from unintentional and intentional attacks and disclosure.

constructor. In programming languages, a method that has the same name as a class and is used to create and initialize objects of that class.

constructor method. See constructor.

conversation. A logical connection between two transaction programs using an LU 6.2 session. Conversations are delimited by brackets to gain exclusive use of a session.

credit. In SETCo., a transaction sent when the merchant needs to return money to the cardholder (via the Acquirer and the Issuer) following a valid capture message, such as when goods have been returned or were defective.

credit reversal. In SETCo., a transaction sent when the information in a previous credit transaction was incorrect or should have never been sent.

cryptographic key. In SETCo., a value which is used to control a cryptographic process, such as encryption or authentication. Knowledge of an appropriate key allows correct decryption or validation of a message.

cryptography. (1) In SETCo., a mathematical process used for encryption or authentication information. (2) The discipline which embodies principles, means, and methods for the transformation of data in order to hide its information content, prevent its undetected modification and unauthorized use, or a combination thereof. (3) The transformation of data to conceal its contents and to prevent one person from forging or modifying another person's messages.

CSP. See commerce service provider.

CyberCash CashRegister. An electronic payment processing service that is provided by CyberCash, Inc. The CyberCash CashRegister enables merchants to accept and process various types of electronic payments for goods or services that are purchased over the Internet.

CyberCash cassette. A payment cassette that provides support for the CyberCash CashRegister.

D

daily batch totals. The Daily Batch Totals report computes the totals for all batches closed on the date specified on the Search window. The totals include all payments and credits made for all payment types.

decryption. In computer security, the process of transforming encoded text or ciphertext into plain text.

derived products. In SETCo., derived products are components that are created from a product that has received a SET Mark license. Derived products must be created from a product that has received the SET Mark, regardless of operating system family. Please refer to the *SET Testing Policies and Procedures* for a complete explanation.

deposit all . Selects all of the order payments displayed for deposit.

deposited amount . The amount deposited for a Payment. The deposited amount can be different than the approved amount.

deposit selected . Selects the order payments that you want to deposit.

digital envelope. (1) In SETCo., a cryptographic technique to encrypt data and send the encryption key along with the data. Generally, a symmetric algorithm is used to encrypt the data and an asymmetric algorithm is used to encrypt the encryption key. (2) In e-commerce, a package of encrypted data and the encryption key.

digital signature. (1) In SETCo., information encrypted with an entity private key, which is appended to a message to assure the recipient of the authenticity and integrity of the message. The digital signature proves that the message was signed by the entity owning, or having access to, the private key. (2) In e-commerce, data that is appended to, or is a cryptographic transformation of, a data unit and that enables the recipient of the data unit to verify the source and integrity of the unit and to recognize potential forgery.

distinguished name. In SET programs, information that uniquely identifies the owner of a certificate.

document type definition (DTD). The rules that specify the structure for a particular class of SGML or XML documents. The DTD defines the structure with elements, attributes, and notations, and it establishes constraints for how each element, attribute, and notation may be used within the particular class of documents. A DTD is analogous to a database schema in that the DTD completely describes the structure for a particular markup language.

DTD. See document type definition.

dual signature. In SETCo., a digital signature that covers two or more data structures by including secure hashes or each data structure in a single encrypted block. Dual signing is done for efficiency, that is, to reduce the number of public key encryption operations.

E

EAR file. An Enterprise Archive file represents a J2EE application that can be deployed in a WebSphere application server. EAR files are standard Java archive files and have the file extension .ear.

e-business. Either (a) the transaction of business over an electronic medium such as the Internet or (b) any organization (for example, commercial, industrial, nonprofit, educational, or governmental) that transacts its business over an electronic medium such as the Internet. An e-business combines the resources of traditional information systems with the vast reach of an electronic medium such as the Internet (including the World Wide Web, intranets, and extranets); it connects critical business systems directly to critical business constituencies--customers, employees, and suppliers. The key to becoming an e-business is building a

transaction-based Web site in which all core business processes (especially all processes that require a dynamic and interactive flow of information) are put online to improve service, cut costs, and sell products.

ECML. See Electronic Commerce Modeling Language.

e-commerce. (1) The exchange of goods and services for payment between the cardholder and merchant when some or all of the transaction is performed via electronic communication. (2) The subset of e-business that involves the exchange of money for goods or services purchased over an electronic medium such as the Internet.

electronic commerce. See e-commerce.

Electronic Commerce Modeling Language (ECML). In e-commerce, a universal format for wallets that streamlines the collection of electronic data for shipping, billing, and payment on a merchant's Web site and thereby enhances the online shopping experience for consumers and merchants. IBM is one of many companies that are collaborating to develop ECML.

encryption. (1) In SETCo., the process of converting information in order to render it into a form unintelligible to all except holders of a specific cryptographic key. Use of encryption protects information between the encryption process and the decryption process (that is, the inverse of encryption), against unauthorized disclosure. (2) In computer security, the process of transforming data into an unintelligible form in such a way that the original data either cannot be obtained or can be obtained only by using a decryption process.

event. In the Tivoli environment, any significant change in the state of a system resource, network resource, or network application. An event can be generated for a problem, for the resolution of a problem, or for the successful completion of a task. Examples of events are: the normal starting and stopping of a process, the abnormal termination of a process, and the malfunctioning of a server.

event listener. In IBM e-commerce, a computer program that waits to be informed of events of interest and acts upon them.

expiry. (1) The certificate expiration date assigned when the certificate was obtained. Certificates are specific to payment types (for example, SET or CyberCash.) (2) Specifies the card expiration date. An expiry value is required for SET protocol. The value is specified as a string and is used on the payment initiation message. For example, 199911 is an expiry value.

Extensible Markup Language (XML). A standard metalanguage for defining markup languages that was derived from and is a subset of SGML. XML omits the more complex and less-used parts of SGML and makes it much easier to (a) write applications to handle document types, (b) author and manage structured information, and (c) transmit and share structured information across diverse computing systems. The use of XML does not require the robust applications and processing that is necessary for SGML. XML is being developed under the auspices of the World Wide Web Consortium (W3C).

F

failover. See fallover.

fallover. In high-availability cluster multiprocessing (HACMP), an active node's acquisition of resources that were previously owned by another cluster node in order to maintain the availability of those resources.

financial institution. (1) In SETCo., an establishment responsible for facilitating customer-initiated transactions or transmissions of funds for the extension of credit or the custody, loan, exchange, or issuance of money, such as a bank or its designate. (2) Within IBM e-commerce, banks, building societies, and credit unions are examples of financial institutions. An institution that provides financial services.

financial network. Within IBM e-commerce, the aggregate of card processors, acquirers, card issuers, and other institutions through which payment card transaction processing is traditionally performed.

firewall. In communication, a functional unit that protects and controls the connection of one network to other networks. The firewall (a) prevents unwanted or unauthorized communication traffic from entering the protected network and (b) allows only selected communication traffic to leave the protected network.

force. Within IBM e-commerce, a WebSphere Commerce Payments verb. An attempt to settle a batch (see 172). If the reconciliation step fails, the batch is still not closed on the WebSphere Commerce Payments (although it may be out of balance or not closed at the financial institution).

FQDN. See fully qualified domain name.

fully qualified domain name (FQDN). In the Internet suite of protocols, the name of a host system that includes all of the subnames of the domain name. An example of a fully qualified domain name is `mycomputer.city.company.com`. See host name.

H

HACMP. See high-availability cluster multiprocessing.

handle. In the AIX operating system, a data structure that is a temporary local identifier for an object. Allocating a handle creates it. Binding a handle makes it identify an object at a specific location.

hardware token. In SETCo., a portable device (for example, smart card, PCMCIA cards) specifically designed to store cryptographic information and possibly perform cryptographic functions in a secure manner.

has been certified. A system that has been inspected and evaluated as fully compliant with the SET protocol by duly authorized parties and process would be said to have been certified.

hash. See root key hash.

heartbeat. In software products, a signal that one entity sends to another to convey that it is still active.

high-availability cluster multiprocessing (HACMP). An application service that enables up to eight RS/6000 servers to access the same data in parallel. This optimizes application execution and scalability and protects against unplanned outages and server downtime.

home page. The initial Web page that is returned by a Web site when a user specifies the uniform resource locator (URL) for the Web site. For example, if a user specifies the URL for the IBM Web site, which is `http://www.ibm.com` , the Web page that is returned is the IBM home page. Essentially, the home page is the entry point for accessing the contents of the Web site. The home page may sometimes be called the ″welcome page″ or the ″front page.″

host. To provide the software and services for managing a Web site.

HostCapture/PostAuth. Within IBM e-commerce, this is a CyberCash concept. One of the three processing models supported by the CyberCash CashRegister service. In particular, the AcquirerProfile field of an account may be set to `HostCapture/PostAuth = 2`, which indicates that the acquirer controls batch processing and a separate deposit request is required to capture the funds after a payment is authorized.

host byte order. The byte order that a central processing unit (CPU) uses to store and process data. This byte order can be big endian or little endian, depending on the particular CPU. Contrast with network byte order.

host name. In the Internet suite of protocols, the name given to a computer. Sometimes, “host name” is used to mean fully qualified domain name; other times, it is used to mean the most specific subname of a fully qualified domain name. For example, if `mycomputer.city.company.com` is the fully qualified domain name, either of the following may be considered the host name:

- `mycomputer.city.company.com`
- `mycomputer`

HTML. See Hypertext Markup Language.

HTTP. See Hypertext Transfer Protocol.

Hypertext Markup Language (HTML). A markup language that conforms to the SGML standard and was designed primarily to support the online display of textual and graphical information that includes hypertext links.

Hypertext Transfer Protocol (HTTP). In the Internet suite of protocols, the protocol that is used to transfer and display hypertext documents.

I

idempotency. (1) A property of a mathematical operation whereby repeating the operation produces no change in the final result. For example, the operation of deducting $25.00 from an account balance is not idempotent, but the operation of setting an account balance to $500.00 is idempotent. (2) In SET Secure Electronic Transaction, a property that enables the sender of a request to repeat the request with a guarantee that the outcome will be the same regardless of whether the request is lost, the response is lost, or the request or response is delayed due to network problems. Idempotency is necessary because the SET protocol works in environments where message delivery is not guaranteed, and when the sender does not receive a response, it cannot determine the cause of the delay. If a SET application does not receive a response in a reasonable amount of time, it resends the message; when the receiving SET application determines that it has already processed that message, it retrieves the previous response and sends that response again.

instrumentation. In application or system software, either (a) monitoring functions that provide performance and other information to a management system or (b) the use of monitoring functions to provide performance and other information to a management system.

identify. To establish the identity.

installment payments. In SETCo., a type of payment transaction negotiated between the merchant and the cardholder which permits the merchant to process multiple authorizations. Cardholder specifies a maximum number of permitted Authorization for paying through installment payments.

integrity. In SETCo., the quality of information or a process that is free from error, whether induced accidentally or intentionally.

interactive. In SETCo., a generic class for a network transport mechanism that is dependent on a logical session being maintained during the message exchange (for example, World Wide Web sessions).

internet. (1) In SETCo., the largest collection of networks in the world, interconnected in such a way as

to allow them to function as a single virtual network. (2) A collection of interconnected networks that use the Internet suite of protocols. The internet that allows universal access is referred to as the Internet (with a capital "I"). An internet that provides restricted access (for example, to a particular enterprise or organization) is frequently called an intranet, whether or not it also connects to the public Internet.

Internet. The worldwide collection of interconnected networks that use the Internet suite of protocols and permit public access.

interoperability. In SETCo., the ability to exchange messages and keys, both manually and in an automated environment, with any other party implementing this standard, provided that both implementations use compatible options of this standard and compatible communications facilities.

interprocess communication (IPC). The process by which programs communicate data to each other and synchronize their activities. Semaphores, signals, and internal message queues are common methods of interprocess communication.

intranet. A private network that integrates Internet standards and applications (such as Web browsers) with an organization's existing computer networking infrastructure.

IP address. The unique 32-bit address that specifies the location of each device or workstation on the Internet. For example, 9.67.97.103 is an IP address.

IPC. See interprocess communication.

issuer. (1) In SETCo., the financial institution or its agent that issues the unique primary account number (PAN) to the cardholder for the payment card brand. (2) In e-commerce, a financial institution that issues payment cards to individuals. An issuer can act as its own certificate authority (CA) or can contract with a third party for the service.

J

J2EE. (Java 2 Enterprise Edition) J2EE is designed to support applications that implement enterprise services for customers, employees, suppliers, partners, and others who make demands on or contributions to the enterprise. This can be a single module or a group of modules packaged into an .ear file with a J2EE application deployment descriptor. J2EE applications are typically engineered to be distributed across multiple computing tiers.

Java. An object-oriented programming language for portable interpretive code that supports interaction among remote objects. Java was developed and specified by Sun Microsystems, Incorporated.

Java Database Connectivity (JDBC). An application programming interface (API) that has the same characteristics as Open Database Connectivity (ODBC) but is specifically designed for use by Java database applications. Also, for databases that do not have a JDBC driver, JDBC includes a JDBC to ODBC bridge, which is a mechanism for converting JDBC to ODBC; it presents the JDBC API to Java database applications and converts this to ODBC. JDBC was developed by Sun Microsystems, Inc. and various partners and vendors.

Java Development Kit (JDK). A software package that can be used to write, compile, debug, and run Java applets and applications.

Java Runtime Environment (JRE). A subset of the Java Development Kit (JDK) that contains the core executables and files that constitute the standard Java platform. The JRE includes the Java Virtual Machine, core classes, and supporting files.

Java Virtual Machine (JVM). A software implementation of a central processing unit (CPU) that runs compiled Java code (applets and applications).

JDBC. See Java Database Connectivity.

JDK. See Java Development Kit.

JRE. See Java Runtime Environment.

JVM. See Java Virtual Machine.

K

keepalive message. In Internet communications, a message sent among nodes when no data traffic has been detected for a given period of time. This communication ensures the vitality of the session by keeping the link "alive."

key. In computer security, a sequence of symbols that is used with a cryptographic algorithm for encrypting or decrypting data. See private key and public key.

key pair. In computer security, a public key and a private key. When the key pair is used for encryption, the sender uses the public key to encrypt the message, and the recipient uses the private key to decrypt the message. When the key pair is used for signing, the signer uses the private key to encrypt a representation of the message, and the recipient uses the public key to decrypt the representation of the message for signature verification. See asymmetric and digital signature.

key ring. In computer security, a file that contains public keys, private keys, trusted roots, and certificates.

L

little endian. A format for storage or transmission of binary data in which the least significant bit (or byte) is placed first. Contrast with big endian.

M

memory leak. A condition in which a computer program allocates memory and does not free (or properly free) this memory. If the program continues to run and is not terminated, it uses large amounts of real memory and eventually runs out of memory.

merchant. In SETCo., a seller of goods, services, and/or other information who accepts payment for these items electronically. The merchant may also provide electronic selling services and/or electronic delivery of items for sale.

merchant chargeback. Within IBM e-commerce, when fraud occurs and a merchant is liable for funds not obtained, a financial institution may issue a merchant chargeback, reclaiming funds previously credited to a merchant's account.

merchant server. (1) In SETCo., a Merchant Server component is a product run by an online merchant to process payment card transactions and authorizations. It communicates with the Cardholder Wallet, Payment Gateway, and Certificate Authority components. (2) In e-commerce, a Web server that offers cataloged shopping.

merchant settings. The settings that a merchant has made for a cassette. In the WebSphere Commerce Payments user interface, the Payment System object displays as Merchant Settings.

MIME. See Multipurpose Internet Mail Extensions.

mirroring. In the AIX operating system, the maintenance of more than one copy of stored data to prevent the loss of data.

Multipurpose Internet Mail Extensions (MIME). An Internet standard for identifying the type of object being transferred across the Internet. MIME types include several variants of audio, graphics, and video.

mutex. A mutual exclusion lock. See mutual exclusion mechanism.

mutual exclusion mechanism. In software, a method for preventing two separately executing pieces of code from interfering with each other's use of a particular data object. For example, if one thread is executing a function that modifies a shared data structure, the application may need to prevent other threads from simultaneously attempting to read the data before the modifications are complete.

N

network. In SETCo., a collection of communication and information processing systems that may be shared among several users.

network byte order. The byte order that a network uses to transmit data. In the Internet, this byte order is always big endian. Contrast with host byte order.

node. See cluster node.

normal mode. In the IBM Payment Gateway, the processing scheme in which a batched SET message is presented in its entirety to the customized exits of the Payment Gateway Application. Contrast with supervisor mode.

number of credits. In SETCo., a credit is a transaction sent when the merchant needs to return money to the cardholder (via the Acquirer and the Issuer) following a valid capture message, such as when goods have been returned or were defective. Credits can be for up to the total amount of all payments associated with an Order. There can be zero or more Credits per Order.

number of payments. In SETCo., a payment is a request by the merchant to the financial institution to approve all or part of an order. In many cases, all the money authorized for collection by the order will be collected in a single payment. Some payment systems may allow the money authorized in one order (that is, one set of payment instructions) to be collected in multiple payments, depending on the business model. There can be zero or more payments per order.

O

ODBC. See Open Database Connectivity.

ODBC bridge. See Java Database Connectivity.

Open Database Connectivity (ODBC). A standard application programming interface (API) for accessing data in both relational and nonrelational database management systems. Using this API, database applications can access data stored in database management systems on a variety of computers even if each database management system uses a different data storage format and programming interface. ODBC is based on the call level interface (CLI) specification of the X/Open SQL Access Group and was developed by Digital Equipment Corporation (DEC), Lotus, Microsoft, and Sybase. Contrast with Java Database Connectivity.

online catalog. In SETCo., shopping on the Internet is simple with online catalogs. Online catalogs are Web pages that display items for sale by an online merchant.

order. An order represents all the instructions and information needed from the consumer (payer) in order for the merchant (payee) to collect money.

order amount. The amount of the order.

order fulfillment. Within IBM e-commerce, merchant systems responsible for shipping or distributing orders for which payment has been received. It is believed that an order fulfillment system would query the WebSphere Commerce Payments to determine what goods are to be shipped.

order search. Search for a single order or group of orders, based on a defined set of characteristics.

out of balance. An unsuccessful attempt was made to balance a batch. All totals do not agree.

P

password. For computer or network security, a specific string of characters entered by a user and authenticated by the system in determining the user's privileges, if any, to access and manipulate the data and operations of the system.

payment. In SETCo., a payment is a request by the merchant to the financial institution to approve all or part of an order. In many cases, all the money authorized for collection by the order will be collected in a single payment. Some payment systems may allow the money authorized in one order (that is, one set of payment instructions) to be collected in multiple payments, depending on the business model.

payment amount. The total payment amount deposited by the merchant for this order.

payment card. (1) In SETCo., a term used to collectively refer to credit cards, debit cards, charge cards, and bank cards issued by a financial institution and which reflects a relationship between the cardholder and the financial institution. (2) In e-commerce, a credit card, debit card, or charge card (a) that is issued by a financial institution and shows a relationship between the cardholder and the financial institution and (b) for which a certificate can be issued from an authenticated certificate authority.

payment cassette. A cassette that implements an electronic payment protocol.

payment gateway. (1) In SETCo., a payment gateway component is a product run by an acquirer or a designated third party that processes merchant authorization and payment messages (including payment instructions from cardholders) and interfaces with private financial networks. (2) In e-commerce, the entity that handles transactions between a merchant and an acquirer.

Payment Gateway Application. In the Payment Gateway Transaction Manager (PGTM), the component that processes SET transactions.

Payment Gateway Transaction Manager (PGTM). In the IBM Payment Gateway, the component that is the application-level routing switch. It provides the protocol-level conversion for managing incoming and outgoing communication, and it provides base services for the intelligent routing of transactions to applications. It manages the communication with merchants and routes transactions among merchants, the Payment Gateway Application, and the acquirer's private financial networks.

payment number. (1) Numeric token. (2) A unique identifier for a particular payment within an order.

payment server. In e-commerce, the electronic equivalent of a cash register that organizes and accepts payment for the goods and services selected for purchase. A payment server uses other components, such as a payment gateway and a payment management system, to complete the financial transactions.

Payment Suite. The brand name for IBM's family of payment products for e-commerce.

PGTM. See Payment Gateway Transaction Manager.

port. In the Internet suite of protocols, a specific logical connector between the Transmission Control Protocol (TCP) or the User Datagram Protocol (UDP) and a higher-level protocol or application. See well-known port.

port number. In the Internet suite of protocols, the identifier for a logical connector between an application entity and the transport service.

primary account number (PAN). In SETCo., the assigned number that identifies the card issuer and cardholder. This account number is composed of an issuer identification number, an individual account number identification, and an accompanying check digit, as defined by ISO 7812–1985.

private key. (1) In SETCo., a cryptographic key used with a public key cryptographic algorithm, uniquely associated with an entity and not made public. This key is used to create digital signatures or to decrypt messages or files. (2) In computer security, a key that is known only to its owner. Contrast with public key. See public key cryptography.

protocol. The meanings of, and the sequencing rules for, requests and responses used for managing a network, transferring data, and synchronizing the states of network components.

public key. (1) In SETCo., a cryptographic key used with a public key cryptographic algorithm, uniquely

associated with an entity publicly available. It is used to verify signatures that were created with he matched private key. Public keys are also used to encrypt messages or files that can only be decrypted using the matched private key. (2) In computer security, a key that is made available to everyone. Contrast with private key. See public key cryptography.

public key cryptography. In computer security, cryptography in which public keys and private keys are used for encryption and decryption.

purge. Within IBM e-commerce, a WebSphere Commerce Payments verb. To remove all associated Payments and Credits from a Batch object, treating it as if it has just been created.

R

random. In SETCo., a value in a set that has equal probability of being selected from the total population of possibilities and is, hence, unpredictable.

realm. In the WebSphere family of products, a database of users, groups, and access control lists. A user must be defined in a realm to access any resource belonging to that realm.

recurring payments. In SETCo., a type of payment transaction initiated by the cardholder that permits the merchant to process multiple authorizations. There are two kinds of recurring payments:

1. multiple payments for a fixed amount
2. repeated billings

refund. Identifies the Credit amount in the smallest denomination of the particular currency used to place the Order.

registration authority. In SETCo., an independent third-party organization that processes payment card applications for multiple payment card brands and forwards applications to the appropriate financial institutions.

reintegration. In high-availability cluster multiprocessing (HACMP), the actions that occur within the cluster when a component that had previously detached from the cluster returns to the cluster. These actions are controlled by the event scripts and when necessary, by manual intervention.

root certificate. In SETCo., the certificate at the top of the certificate hierarchy. See certificate chain.

root key hash. In SET programs, a hexadecimal value that is used to verify the validity of a root certificate. The hash value is published for a consumer to use when the software does not recognize the root certificate.

rotating. In high-availability cluster multiprocessing (HACMP), pertaining to a cluster configuration in which the cluster node with the highest priority for a particular resource acquires the resource if the primary node fails and retains the resource even upon reintegration of the primary node into the cluster. Contrast with cascading and concurrent.

RS/6000. A family of workstations and servers based on IBM's POWER architecture. They are primarily designed for running multi-user numerical computing applications that use the AIX operating system.

S

sale. (1) In the credit card world, a sale occurs when a transaction is authorized and marked for capture all at once rather than using a two-step process. (2) Within IBM e-commerce, a WebSphere Commerce Payments user interface verb. It means a simultaneous Approve and Deposit.

sale all. Selects all orders displayed to approve and move the associated payment directly into deposited state. The sale function automatically performs an approve and a deposit on your payment.

sale selected. Selects the orders that you want to approve and move the associated payment directly into deposited state. The sale function automatically performs an approve and a deposit on your payment.

sales transaction. In SETCo., a payment authorization transaction that allows a merchant to authorize a transaction and request payment in a single message to the acquirer.

Secure Electronic Transaction. See SET Secure Electronic Transaction.

Secure Sockets Layer (SSL). (1) In SETCo., Secure Socket Layer (SSL) (developed by Netscape Communications Company) is a standard that encrypts data between a Web browser and a Web server. SSL does not specify what data is sent or encrypted. In an SSL session, all data sent is encrypted. (2) A security protocol that provides communication privacy. SSL enables client/server applications to communicate in a way that is designed to prevent eavesdropping, tampering, and message forgery. SSL was developed by Netscape Communications Corp. and RSA Data Security, Inc.

server. In SETCo., a functional unit that provides services to one or more clients over a network. Examples include a file server, a print server, and a mail server.

servlet. An application program, written in the Java programming language, that is executed on a Web server. A reference to a servlet appears in the markup for a Web page, in the same way that a reference to a graphics file appears. The Web server executes the

servlet and sends the results of the execution (if there are any) to the Web browser. Contrast with applet.

SET. See SET Secure Electronic Transaction.

SET logo. In SETCo., the SET logo or SET Mark is your assurance that the merchant is using software that has successfully completed the SET Software Certification test.

SET cassette. A payment cassette that provides support for the SET protocol.

SET Secure Electronic Transaction. (1) In SETCo., the SET Secure Electronic Transaction™ protocol is an open industry standard developed for the secure transmission of payment information over the Internet and other electronic networks. (2) A specification for securing payment card transactions over open networks such as the Internet. SET was developed by Visa, MasterCard, IBM, and other technology companies.

settle. Within IBM e-commerce, a WebSphere Commerce Payments verb. An attempt to close a Batch object and transfer funds. As part of the settling procedure, there may be some reconciliation or balancing steps (depending on the cassette and financial institution policy) to ensure that the merchant and financial institution agree on the funds being transferred. If the reconciliation step fails, the batch may remain in an open state.

settle all. Settles all batches displayed. The batches can then submit payments and refunds for processing by a payment processor.

settle batches. Settle batches is used to submit batches (payments and refunds) for processing by a payment processor. You can choose to settle one Batch, or multiple Batches.

settle selected. Settles the batches you selected. The selected batches can then submit payments and refunds for processing by a payment processor.

sibling. In SETCo., sibling products are components which, by virtue of being within the same operating system family, are closely related to baseline products. Siblings must be of the same operating system family as the baseline product from which they were created, with identical functionality. Refer to the *SET Testing Policies and Procedures* for a complete explanation.

SMIT. See System Management Interface Tool.

socket. An endpoint provided by the transport service of a network for communication between processes or application programs.

socks-enabled. Pertaining to TCP/IP software, or to a specific TCP/IP application, that understands the socks protocol. “Socksified” is a slang term for socks-enabled.

socksified. See socks-enabled.

socks protocol. A protocol that enables an application in a secure network to communicate through a firewall via a socks server.

socks port. The port on which the Socks server is listening.

socks server. A circuit-level gateway that provides a secure one-way connection through a firewall to server applications in a nonsecure network.

SSL. See Secure Sockets Layer.

stack. A slang term for the set of protocols that comprise TCP/IP. The preferred term is TCP/IP.

supervisor. Can perform all payment processing functions for the merchant.

supervisor mode. In the IBM Payment Gateway, the processing scheme in which a batched SET message is presented as a series of individual requests to the customized exits of the Payment Gateway Application. Contrast with normal mode.

System Management Interface Tool (SMIT). An interface tool of the AIX operating system for installing, maintaining, configuring, and diagnosing tasks.

T

TEC. See Tivoli Enterprise Console.

terminal capture. Within IBM e-commerce, a CyberCash concept. One of the three processing models supported by the CyberCash CashRegister service. In particular, the AcquirerProfile field of an account may be set to `Terminal Capture = 3` , which indicates that the merchant controls batch processing.

thread. A stream of computer instructions that is in control of a process. A multi-threaded process begins with one stream of instructions (one thread) and may later create other instruction streams to perform tasks.

thread pool. The threads that are being used by or are available to a computer program.

time approved. The date and time that this Payment entry was created.

time opened. The time that the batch was created.

time ordered. The time that the order entry was created.

Tivoli Enterprise Console (TEC). A Tivoli product that collects, processes, and automatically initiates corrective actions for system, application, network, and database events; it is the central control point for events from all sources. The Tivoli Enterprise Console provides a

centralized, global view of the network computing environment; it uses distributed event monitors to collect information, a central event server to process information, and distributed event consoles to present information to system administrators.

Tivoli GEM. See Tivoli Global Enterprise Manager.

Tivoli Global Enterprise Manager (Tivoli GEM). A Tivoli product that allows system administrators to graphically monitor, control, and configure applications residing in distributed and host (S/390) environments and to use the concept of business systems management to organize related components, thereby providing a business perspective for management decisions. Tivoli Global Enterprise Manager gives information technology staff a logical view of the computing environment; this view shows, at a glance, the status of the multiple applications that comprise the enterprise's business system, including application components, the relationships among and between components, and the flow of data between the applications. By providing this view from a business perspective, Tivoli Global Enterprise Manager enables system administrators to quickly make determinations about the business impact of any component failure. Addressing technology problems from the business perspective greatly improves the effectiveness of system administrators and provides a higher level of service to users.

Tivoli Inventory. A Tivoli product that enables system administrators to gather hardware and software information for a network computing environment. It scans the managed resources and stores inventory information in the configuration repository.

Tivoli management software. The overall descriptor for software from Tivoli Systems Inc., which includes Tivoli Enterprise software (for systems management in a large organization), Tivoli IT Director (for systems management in a small or medium organization), and Tivoli Cross-Site (for the management of e-commerce systems). Tivoli management software enables organizations to centrally manage their computing resources (including the critical applications that drive business performance and profits) in a simple and straightforward manner.

Tivoli Ready. Pertaining to a product that has passed rigorous product certification testing by Tivoli Systems Inc. to ensure that the product delivers turnkey (or ″out-of-the-box″) integration with Tivoli management software. A product that has passed this certification testing carries the Tivoli Ready logo.

transaction. In SETCo., a sequence of one or more related messages.

trust chain. In SETCo., a synonym for certificate chain. See 164.

trusted root. In the Secure Sockets Layer (SSL), the public key and associated distinguished name of a certificate authority (CA).

U

uniform resource locator (URL). (1) A sequence of characters that represent information resources on a computer or in a network such as the Internet. This sequence of characters includes (a) the abbreviated name of the protocol used to access the information resource and (b) the information used by the protocol to locate the information resource. For example, in the context of the Internet, these are abbreviated names of some protocols used to access various information resources: `http` , `ftp` , `gopher` , `telnet` , and `news` ; and this is the URL for the IBM home page: `http://www.ibm.com` . (2) The address of an item on the World Wide Web. It includes the protocol followed by the fully qualified domain name (sometimes called the host name) and the request. The Web server typically maps the request portion of the URL to a path and file name. For example, if the URL is `http://www.networking.ibm.com/nsg/nsgmain.htm` , the protocol is `http` ; the fully qualified domain name is `www.networking.ibm.com`; and the request is `/nsg/nsgmain.htm`.

URL. See uniform resource locator.

user exit routine. A user-written routine that receives control at predefined user exit points. User exit routines can be written in assembler or a high-level language.

V

virtual sales slip. In SETCo., detailed information on a financial transaction which is generated by the merchant's online store and downloaded to your digital wallet. Typical items contained in the virtual sales slip are confirmation of your order, shipping details, tax (if applicable), and total amount of sale.

virtual store. An interactive simulation of a store on the World Wide Web.

void payment. Within IBM e-commerce, a verb meaning to nullify or cancel a payment operation (that is, to make it as if it never happened).

W

wallet. In the IBM Payment Suite, software that enables a user to make approved payments to authenticated merchants over public networks and to manage payment card accounts and purchases.

WAR file. A Web Archive (WAR) file is a Java archive file used to store one or more of the following: servlets; JavaServer Pages (JSP) files; utility classes; static

documents (such as HTML files, images and sound); client-side applets, beans and classes; descriptive meta-information. Its standard file extension is .war. WAR files are used to package Web modules.

Web. See World Wide Web.

Web browser. (1) Within IBM e-commerce, software running on the cardholder processing system that provides interface to public data networks. (2) A client program that initiates requests to a Web server and displays the information that the server returns.

Web browser plug-in. In SETCo., software installed on the cardholder's computer used to add functions to the Web browser.

webmaster. The person who is ultimately responsible for managing and maintaining a particular Web site.

Web page. Any document that can be accessed by a uniform resource locator (URL) on the World Wide Web. Contrast with home page.

Web server. A server that is connected to the Internet and is dedicated to serving Web pages.

Web site. A Web server that is managed by a single entity (an organization or an individual) and contains information in hypertext for its users, often including hypertext links to other Web sites. Each Web site has a home page. In a uniform resource locator (URL), the Web site is indicated by the fully qualified domain name. For example, in the URL `http://www.networking.ibm.com/nsg/nsgmain.htm` , the Web site is indicated by `www.networking.ibm.com` , which is the fully qualified domain name.

WebSphere. Pertaining to a family of IBM software products that provide a development and deployment environment for basic Web publishing and for transaction-intensive, enterprise-scale e-business applications.

well-known port. In the Internet suite of protocols, one of a set of preassigned protocol port numbers that address specific functions used by transport-level protocols such as the Transmission Control Protocol (TCP) and the User Datagram Protocol (UDP). The File Transfer Protocol (FTP) and the Simple Mail Transfer Protocol (SMTP), for example, use well-known port numbers.

World Wide Web (WWW). A network of servers that contain programs and files. Many of the files contain hypertext links to other documents available through the network.

WWW. See World Wide Web.

X

XML. See Extensible Markup Language.

Numerics

2KP transaction. A SET transaction in which the cardholder messages are unsigned and two key pairs (one for the merchant and one for the payment gateway) are used for encryption.

3KP transaction. A SET transaction in which the cardholder messages are unsigned and three key pairs (one for the merchant, one for the payment gateway, and one for the cardholder) are used for encryption.

Index

Special Characters

A

B

C

D

E

F

H

I

J

K

L

M

N

O

P

Q

R

S

T

U

W

X

Z

Readers' Comments — We'd Like to Hear from You

IBM WebSphere Commerce Payments for Multiplatforms
Programmer's Guide and Reference
Version 3.1

Overall, how satisfied are you with the information in this book?

	Very Satisfied	Satisfied	Neutral	Dissatisfied	Very Dissatisfied
Overall satisfaction	☐	☐	☐	☐	☐

How satisfied are you that the information in this book is:

	Very Satisfied	Satisfied	Neutral	Dissatisfied	Very Dissatisfied
Accurate	☐	☐	☐	☐	☐
Complete	☐	☐	☐	☐	☐
Easy to find	☐	☐	☐	☐	☐
Easy to understand	☐	☐	☐	☐	☐
Well organized	☐	☐	☐	☐	☐
Applicable to your tasks	☐	☐	☐	☐	☐

Please tell us how we can improve this book:

Thank you for your responses. May we contact you? ☐ Yes ☐ No

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

Name _____ Address _____

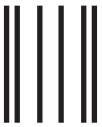
Company or Organization _____

Phone No. _____

Readers' Comments — We'd Like to Hear from You

Cut or Fold
Along Line

Fold and Tape **Please do not staple** Fold and Tape



NO POSTAGE
NECESSARY
IF MAILED IN THE
UNITED STATES

BUSINESS REPLY MAIL

FIRST-CLASS MAIL PERMIT NO. 40 ARMONK, NEW YORK

POSTAGE WILL BE PAID BY ADDRESSEE

IBM Corporation
Software Reengineering
Department G7IA / Bldg 503
P.O. Box 12195
Research Triangle Park, NC
27709-9990

Fold and Tape **Please do not staple** Fold and Tape

Cut or Fold
Along Line

Printed in U.S.A.