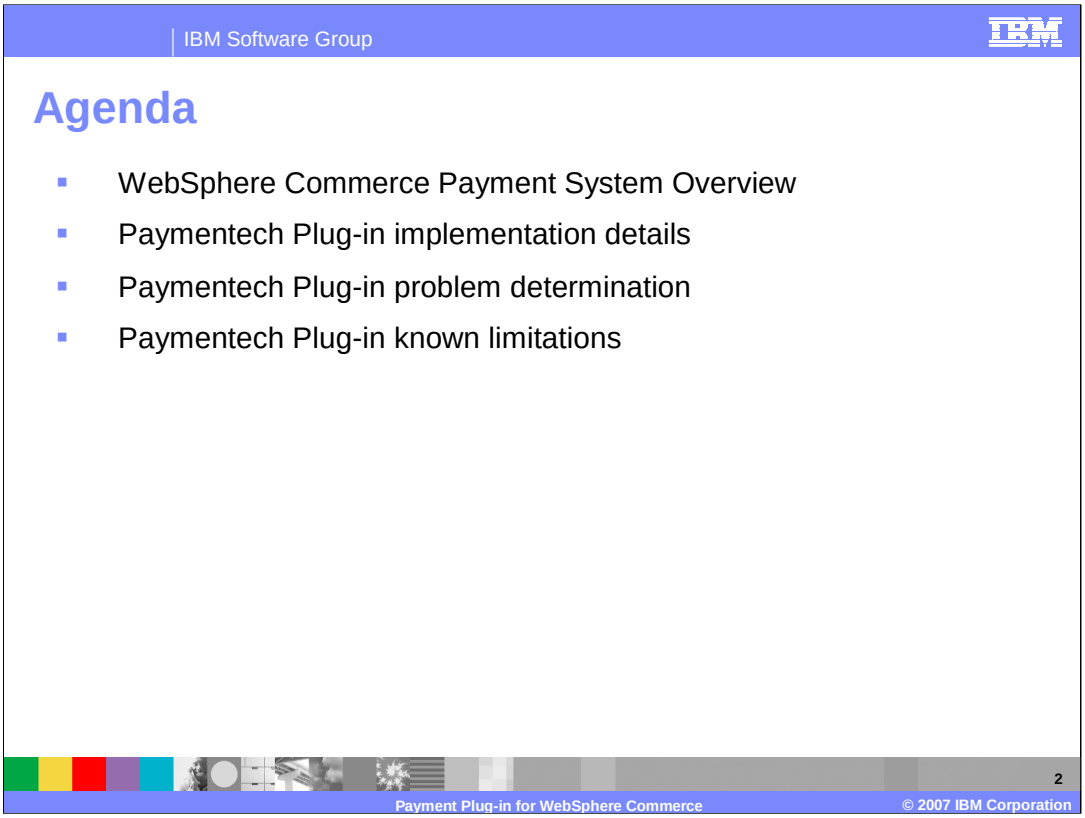
Welcome to the WebSphere Commerce Feature Pack 2, Paymentech Plug-in presentation.



This presentation discusses the following topics:

The overview briefly reviews WebSphere Commerce V6 payment processing system. Then it introduces what the Paymentech Plug-in is and what credit card types and currencies are supported.

The Paymentech Plug-in implementation details section discusses the protocol used by the Paymentech Plug-in, how the Paymentech plug-in handles on-line processing, batch processing and the Paymentech Plug-in Controller enhancement for batch processing support. At end of this session, Paymentech Plug-in installation and configuration steps are covered.

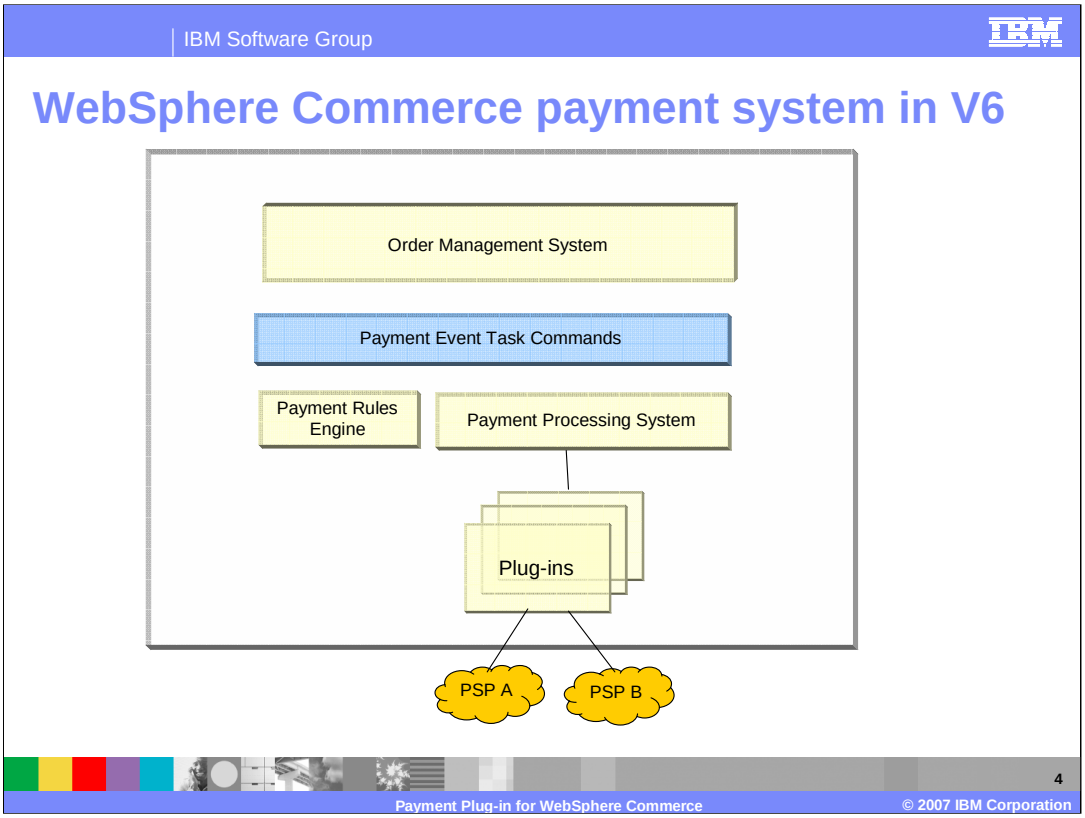
The Paymentech Plug-in problem determination section discusses how to retrieve back end transaction IDs and return codes and how to recover from transaction failure.

The last session covers Paymentech Plug-in known limitations within Feature Pack 2.

Section

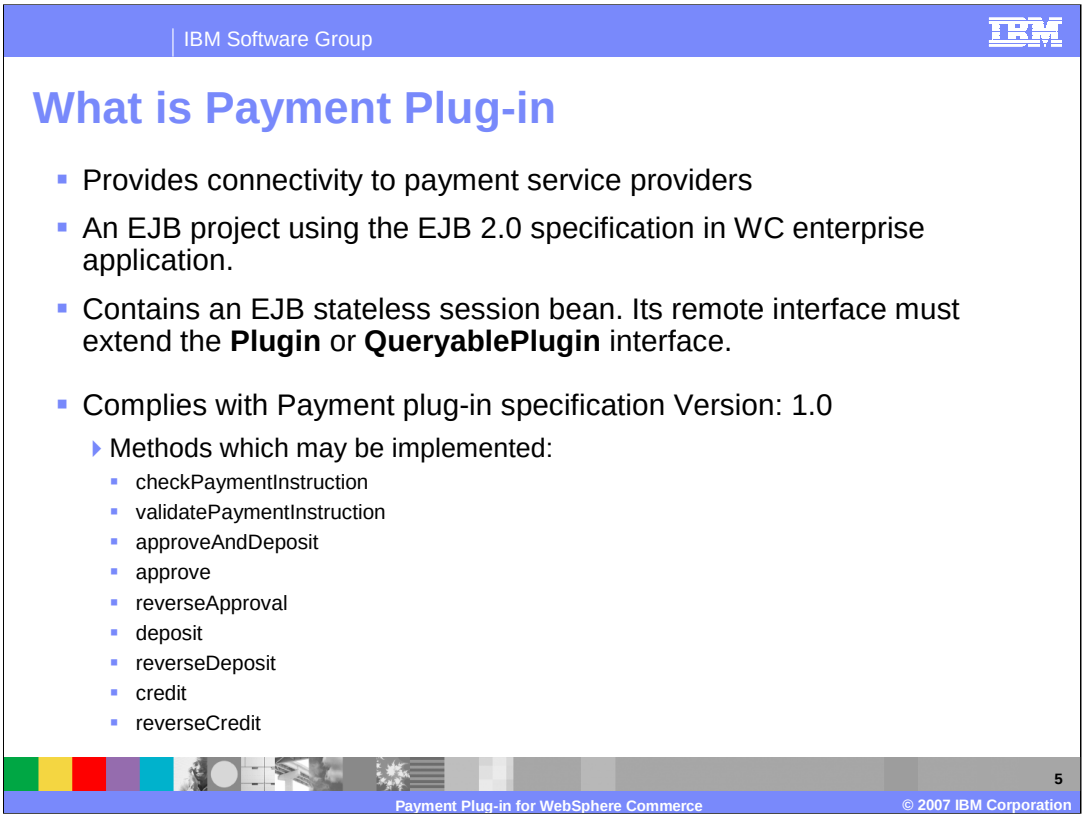
WebSphere Commerce Payment System Overview

This section discusses the overview of the WebSphere Commerce Payment system.



WebSphere Commerce payment system contains 5 components: **Order Management System, Payment Event Task Commands, Payment Rules Engine, Payment Processing System and Payment Plug-ins.**

The **Order Management subsystem** delegates any payment processing and payment decisions to the Payment Rules sub-component. It generates payment events, but it does not contain any payment logic. **Payment Event Task Commands** call the payment rules component to determine payment actions. **The Payment Rules Engine** evaluates payment rules and determines what payment actions to run for a specific payment event. The configurable payment rules determine what payment actions should be executed. By using payment rules, it allows supporting multiple payment instructions per order and multiple releases per order. The Payment Rules Engine is clone-able because it is a part of the WebSphere Commerce enterprise application. **The Payment Processing System** routes payment actions to payment plug-ins, manages the persistence and state transitions of payment and financial transaction records. The Payment Processing System is part of the WebSphere Commerce enterprise application, so it is clone-able. **The Payment Plug-in** provides the integration point to a back-end Payment Service Provider (PSP).



A payment plug-in is a self-contained software component that serves as a proxy for a payment back-end system. To validate a payment, the payment plug-in receives data about a financial transaction from the payment system, then sends a request to the payment back-end system. After receiving a response from the back-end system, the payment plug-in decides what data should be stored and what appropriate information should be returned to WebSphere Commerce Payments. WebSphere Commerce Payments then performs the database operations necessary to record all the financial transactions for a payment plug-in.

In WebSphere Commerce, a payment plug-in is an EJB module that is compliant with the EJB 2.0 specification. This EJB module must contain an EJB stateless session bean and its remote interface must extend the **Plugin** or **QueryablePlugin** interface. The EJB must also comply with the WebSphere Commerce Payment plug-in specification V1.0

In the plug-in specification, the interface defines methods which handle the following financial transactions:

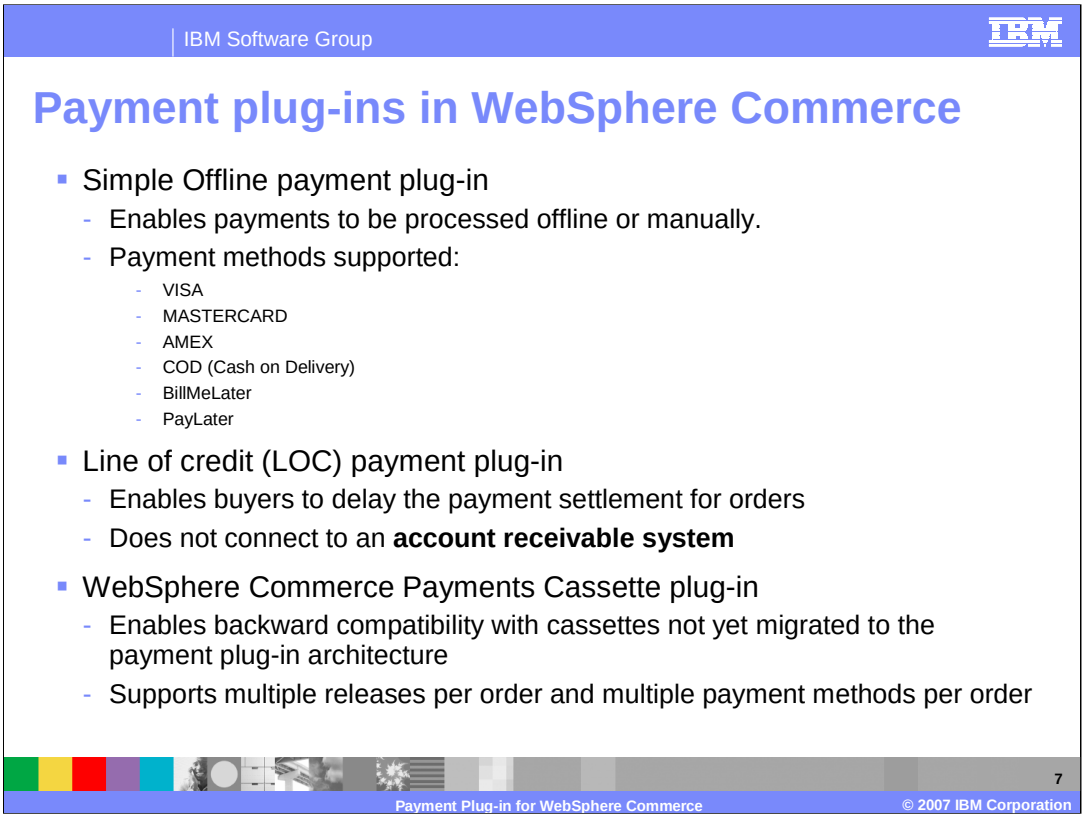
1. Payment approval (or authorization),
2. Payment deposit (or capture)
3. Credit (or refund) or
4. Reversals for the above transactions.

Based on the payment protocol and the back-end system capability, some or all of these methods need to be implemented.

What is Payment Plug-in (cont.)

- Plug-in deployment descriptor PluginDeployment.xml
  - Location: WC_eardir/xml/config/payments/**ppc**/plugins/<plug-in-name>/PluginDeployment.xml
  - Standard properties that a plug-in must have:
    - **jndi** – JNDI name for stateless session bean
    - **home** - The fully qualified name of the home class for the stateless session bean
    - **name** - The name of the plug-in implementation
    - **Version** - The version of the plug-in implementation
    - **Vendor** - The vendor providing the plug-in implementation
    - **supportIndependentCredit** - Indication of whether the plug-in supports independent credits
    - **virtualTerminal** - The URL of the payment back-end system administration user interface.

Each payment plug-in contains a plug-in deployment descriptor PluginDeployment.xml. This file is in the **config** directory under the Payment Plug-in Controller (PPC). It defines several standard properties such as **JNDI** for name of the stateless session bean and **home** for the fully qualified name of the home class for the stateless session bean. These two properties will be used by PPC to call the plug-in.



WebSphere Commerce provides 3 payment plug-ins: the Simple Offline Payment plug-in**,** the Line of credit (LOC) payment plug-in and the WebSphere Commerce Payments Cassette plug-in. The first two plug-ins are offline plug-ins, the third one is an online plug-in.

The **Simple Offline Payment plug-in** processes payments offline or manually. Offline payments do not involve any direct communication with a payment back-end system. Instead, the Simple Offline plug-in records events that have already happened outside of WebSphere Commerce. Transactions are recorded and maintained in the WebSphere Commerce database. The WebSphere Commerce Payment system performs the database operations necessary to record all the payment transactions for this plug-in.

The **Line of credit (LOC) payment plug-in** enables merchants to use lines of credit as a payment method. It enables buyers to delay the payment settlement for orders and is intended to integrate with your accounts receivable system. The LOC plug-in simulates the function of an actual line of credit, but by default does not connect to an accounts receivable system. To connect to an accounts receivable system, you must either modify the LOC plug-in or write your own version of the plug-in. The plug-in can be used as an example when developing your own plug-ins.

The **Payments Cassette plug-in** allows you to continue using the Payment Manager for payment processing services along with any of your existing cassettes. This is intended to be used for backwards compatibility while migrating to the new payment architecture. It is an optional component that can be selected during WebSphere Commerce installation. If you want to use this plug-in, you must have selected WebSphere Commerce Payments during installation.

What is Paymentech Plug-in

- Based on payment plug-in specification V1.0
- Performs online payment authorization processing
- Performs batch processing for deposit and credit transactions
- Uses Paymentech as service provider
- Communicates with Paymentech gateways directly
- Uses Frame Relay TCP/IP connection between plug-in and gateway
- Replaces WCPayments plug-in with the Paymentech Cassette

The Paymentech plug-in is a WebSphere Commerce payment plug-in whose implementation is based on payment plug-in specification V1.0. It uses online processing for payment authorization and batch processing for deposit or credit transactions. The Paymentech plug-in uses Paymentech Inc. as the service provider and communicates with the Paymentech gateways directly using the TCP/IP protocol.

Paymentech Plug-in payment methods

Supported payment methods by Paymentech	Certified payment methods by Paymentech plug-in
Visa	Visa
MasterCard	MasterCard
American Express	American Express
Discover	Discover
Novus	
Diners	
Carte Blanche	
JCB	
Private Label	

Country/Region	Currency name	ISO 4217 Code
United States	United States dollar	USD
Canada	Canadian dollar	CAD
Japan	Japanese yen	JPY
Euro-supported countries: Austria Belgium Finland France Germany Ireland Italy Luxembourg Netherlands Portugal Spain	European euro currency	EUR
United Kingdom	British pound	GBP
Australia	Australian dollar	AUD

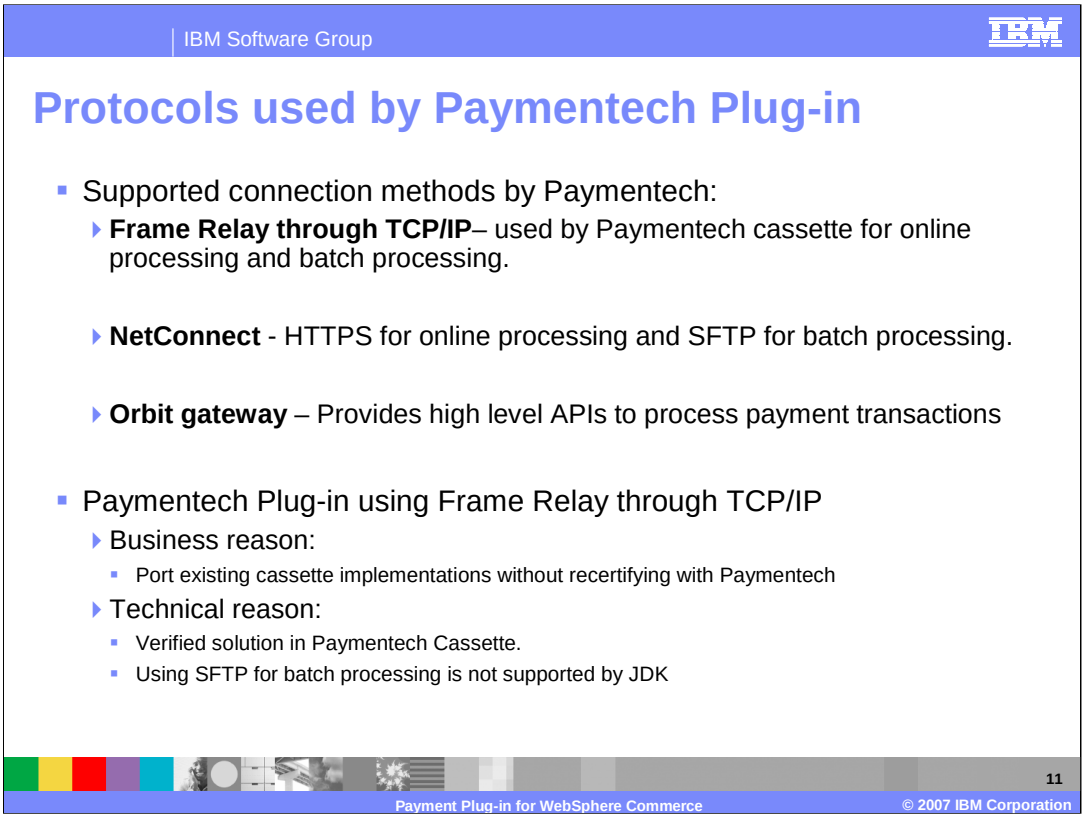
The Paymentech Plug-in only supports credit card transactions. The following credit card brands are certified with the Paymentech plug-in: VISA, Master Card, American Express, and Discover. However, all other credit card brands supported by Paymentech are also supported by Paymentech plug-in. If you need to use other payment methods, then you need to test them yourself.

The certified currencies for the Paymentech plug-in are the U.S. Dollar, the Canadian Dollar, the Euro, the British Pound, the Japanese Yen and the Australian Dollar.

Section

Paymentech Plug-in implementation details

This section discusses the Paymentech Plug-in implementation details.

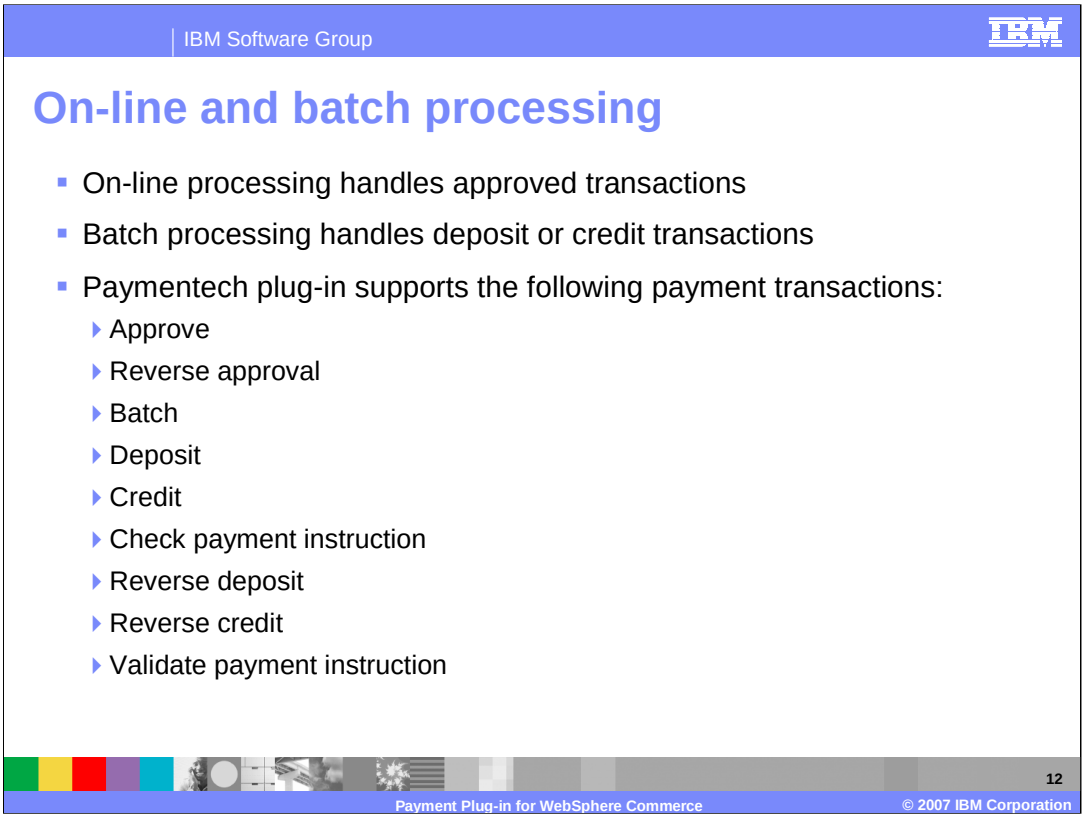


Paymentech Inc. provides three options to connect to Paymentech gateways : **Frame Relay through TCP/IP**, **NetConnect** and **Orbit gateway**.

Frame Relay through TCP/IP is the connection method used by WebSphere Paymentech cassette for online processing and batch processing. For the **NetConnect** implementation, Paymentech requires the use of the HTTPS protocol for online processing and the SFTP protocol (secured FTP) for batch processing. For the **Orbit gateway** implementation, the high level API's provided by Paymentech must be used to process payment transactions.

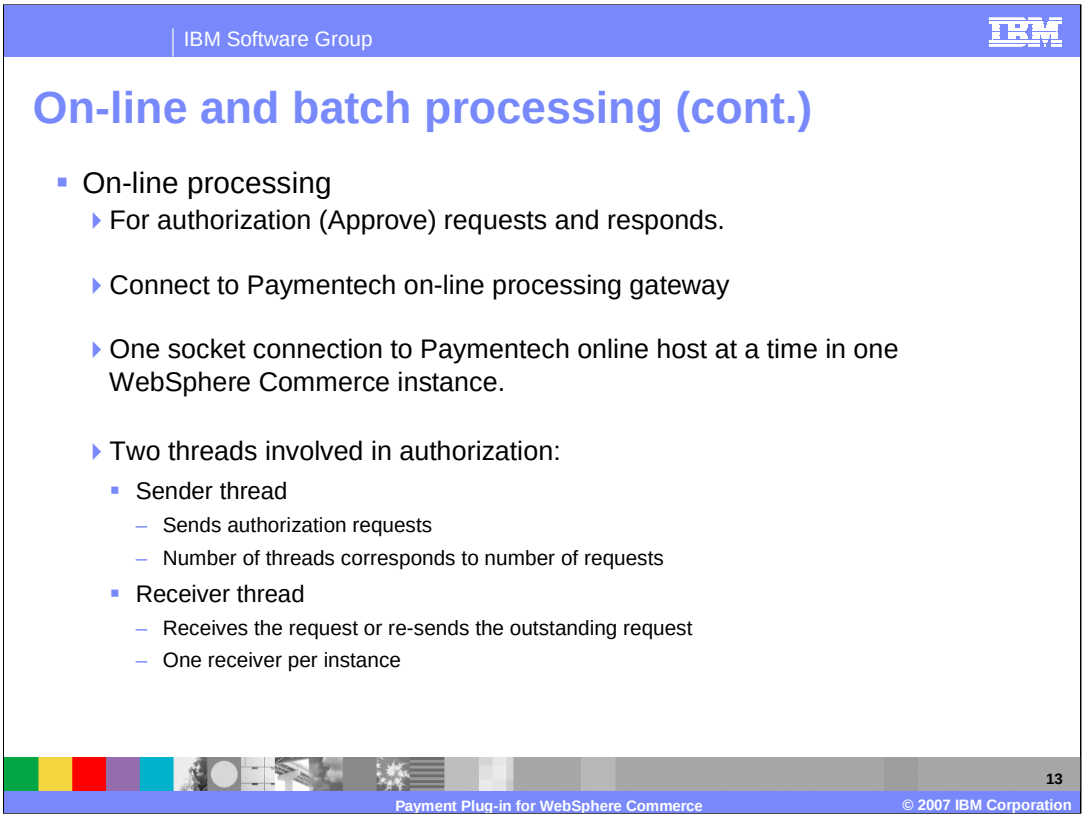
Paymentech Plug-in uses Frame Relay through TCP/IP as its implementation. The decision to choose the TCP/IP implementation is based on following facts:

1. WebSphere Commerce Paymentech cassette uses TCP/IP socket connections both in online authorization processing and batch processing. If you are using the WebSphere Commerce Paymentech cassette, you can easily migrate to the new Paymentech plug-in implementation without recertifying with Paymentech Inc.
2. TCP/IP is a stable and acceptable solution that has been verified with the WebSphere Commerce Paymentech Cassette. Using other implementation options will increasing the development time. NetConnect uses SFTP for batch processing and SFTP is not supported by the JDK. You should use vendor software if you are using NetConnect.



The Paymentech gateway does not allow online processing of all transactions. Approved transactions are handled by the online processing gateway. Deposit or credit transactions are handled by the batch processing gateway.

The following payment transactions are supported by the Paymentech plug-in: Approve, Reverse approval, Batch, Deposit, Credit, Check payment instruction, Reverse deposit, Reverse credit and Validate payment instruction.

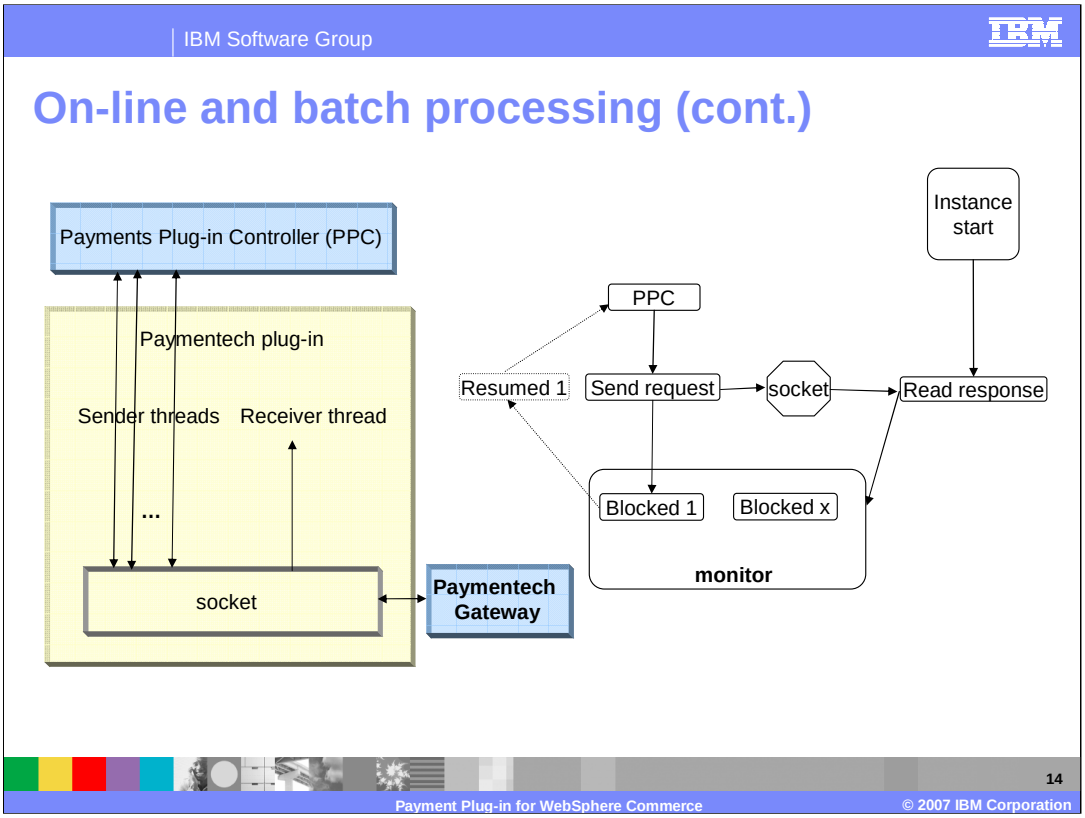


On-line processing is used for authorizing requests and responses. It connects to the Paymentech on-line processing gateway. To fulfill the best interchange rate, the Paymentech plug-in implementation allows only one socket connection to the Paymentech online host at a time in one WebSphere Commerce instance.

There are two types of threads involved in authorization:

1. A sender thread sends authorization transaction requests to Paymentech. The number of threads equals the number of requests.

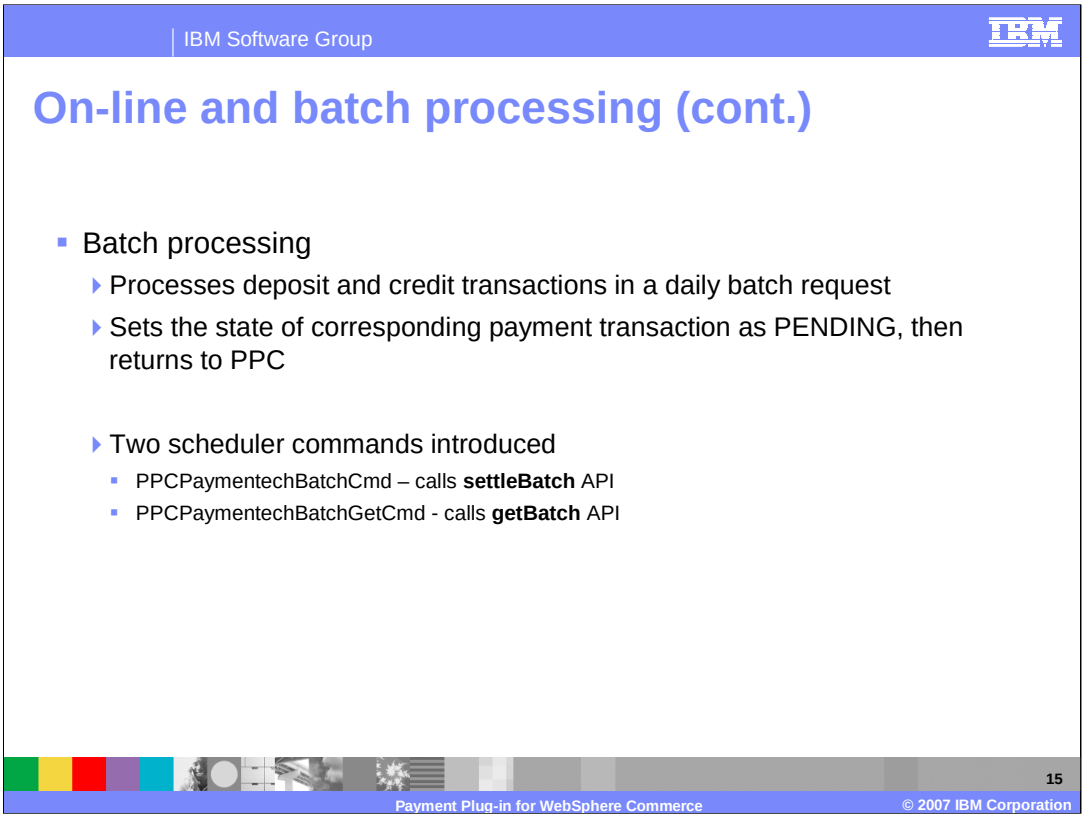
2. A receiver thread receives the requests or re-sends the outstanding requests. There is only one receiver thread per commerce instance.



The left side of the chart shows these two types of threads in the Paymentech plug-in. The sender thread is actually an API thread. It is an EJB bean thread called by the Payment Plug-in Controller. In order to send a complete request, only **one** thread is allowed to send on the socket at a time. After sending, the API thread remains blocked to wait on the specified monitor to be notified by the receiver thread. The receiver thread is responsible for coordinating the reading of Paymentech responses and for resending outstanding requests.

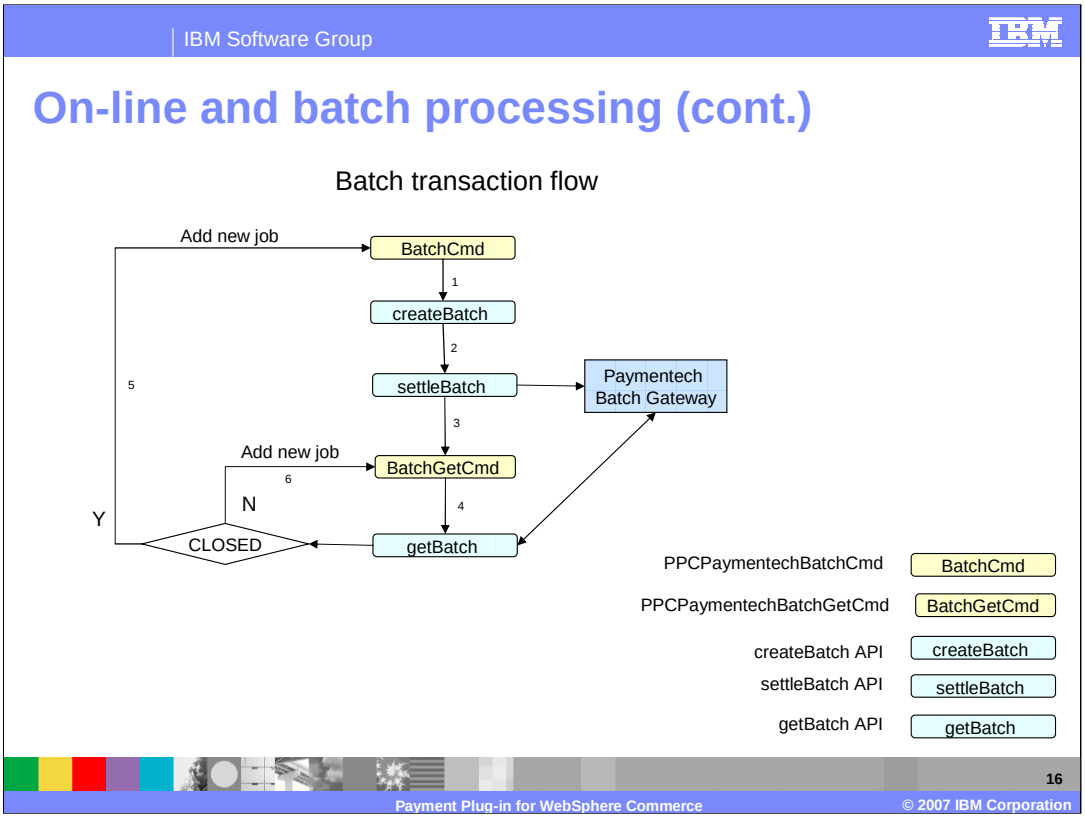
The work flow of these threads is shown on the right hand side of the chart:

1.The receiver thread is started when the WebSphere Commerce instance is started. This receiver thread stays alive and circularly tries to read response data from the socket. This thread stays alive until told to stop.
2. The Payments Plug-in Controller calls the Paymentech plug-in to process authorization transactions.
3. If all request data is successfully written to the socket, the current sender thread is blocked, it waits for the specified monitor to be notified by the receiver thread.
4. If the receiver thread successfully read the response of a specified authorization transaction, it will notify the blocked sender thread corresponding to the authorization request.
5. The blocked thread resumes, it parses the response message read from the socket by the receiver thread and returns the transaction processing result to the Payment Plug-in Controller.



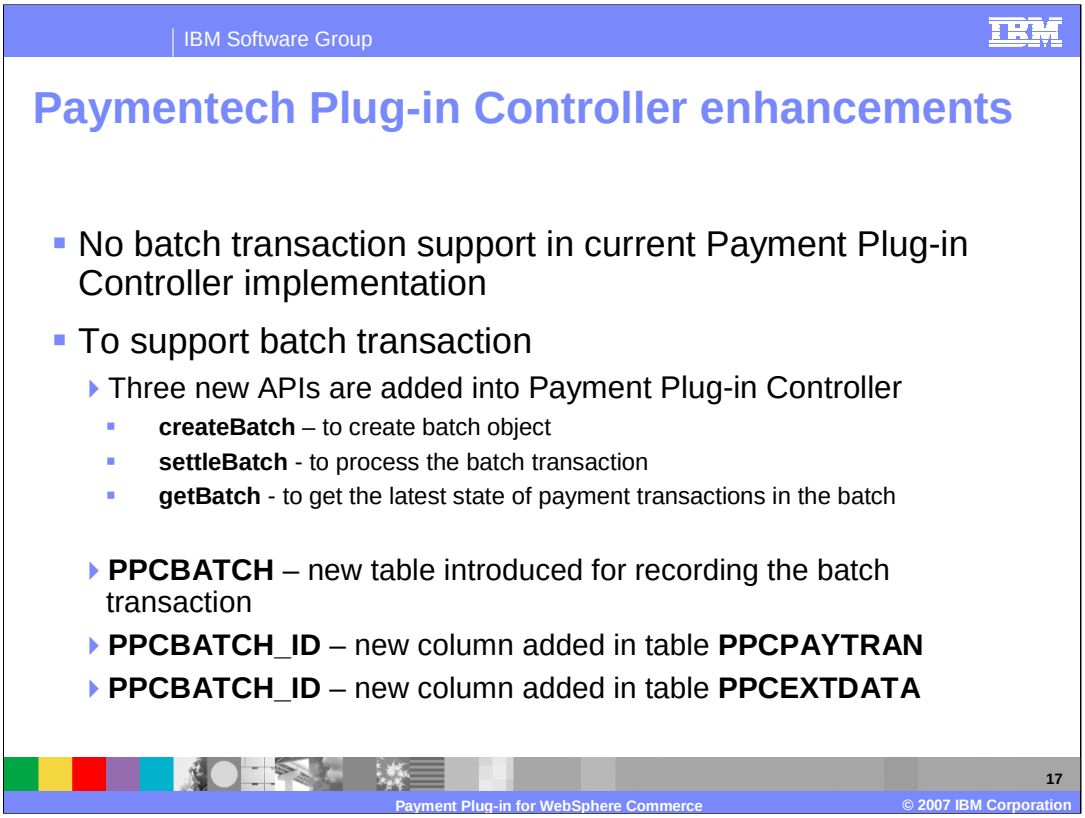
The Paymentech gateway processes deposit and credit transactions in a batch request. The batch is typically sent daily such that when the Payment Plug-in Controller calls the plug-in deposit or credit method, the plug-in does not send the request to the Paymentech gateway. Instead, it only sets the state of the corresponding payment transaction as **PENDING** then returns to Payment Plug-in Controller.

In order to support batch processing, two new scheduler commands are introduced: PPCPaymentechBatchCmd and PPCPaymentechBatchGetCmd. PPCPaymentechBatchCmd calls **settleBatch** API to send batch transaction out. PPCPaymentechBatchGetCmd calls **getBatch** API to get the batch transaction results from Paymentech.



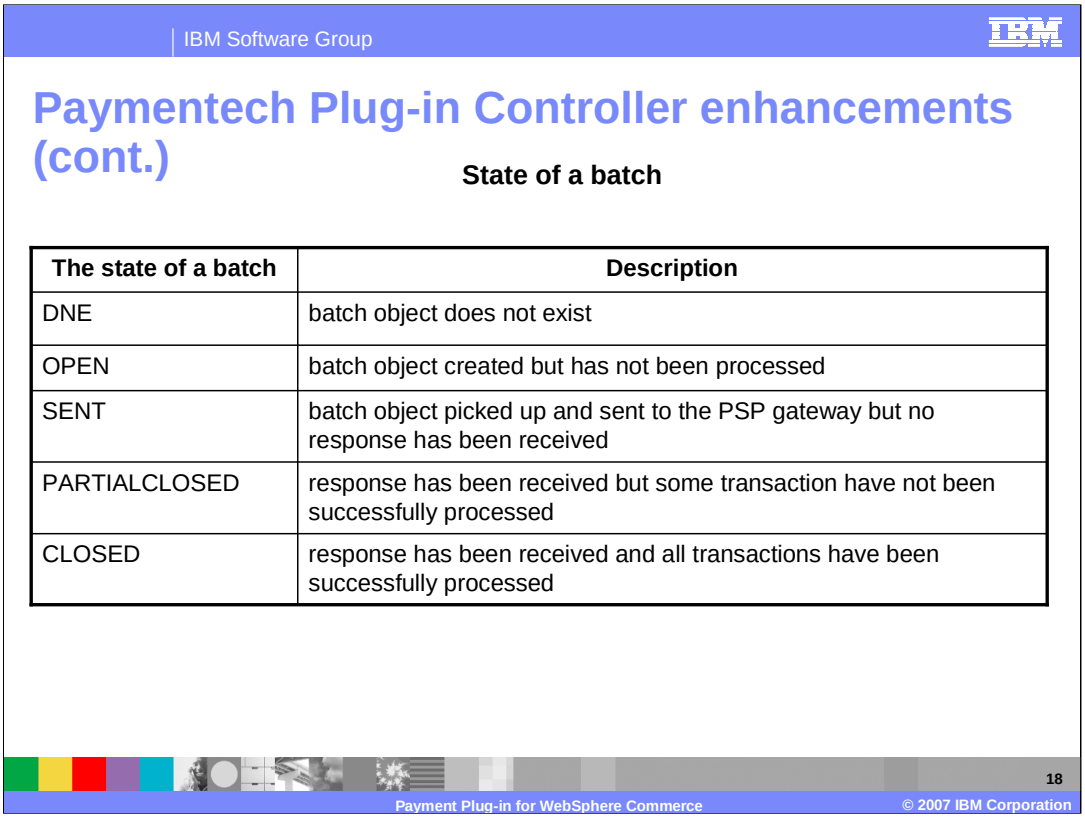
This chart shows how these two scheduled commands work together for batch processing:

If the Paymentech Plug-in is used, the PPCPaymentechBatchCmd runs for each merchant configuration. At a scheduled time, the **PPCPaymentechBatchCmd** calls the Payment Plug-in Controller's **createBatch** API to create a new batch. After a new batch is created, **createBatch** calls the **settleBatch** API to send the batch to the Paymentech batch gateway. After the batch request is sent successfully, the socket is closed. At that point, another scheduled job PPCPaymentechBatchGetCmd is added to the system. At a scheduled time, the **PPCPaymentechBatchGetCmd** calls the **getBatch** API to get the results of the batch requests from the Paymentech batch gateway. If the returned batch is in the **CLOSED** or **PARTIALCLOSED** state, a new PPCPaymentechBatchCmd is added. If the returned batch is still in the **SENT** state, a new PPCPaymentechBatchGetCmd job will be added for this batch.



WebSphere Commerce V6.0 does not support batch transactions. In Feature Pack 2, three new APIs have been added to the Paymentech Plug-in Controller, they are **createBatch, settleBatch and getBatch.** These three APIs are used to create batch objects, process the batch transactions and retrieve the state of payment transactions in the batch.

Database tables have also been changed. The **PPCBATCH** table has been introduced for recording the batch transaction. A new column, **PPCBATCH_ID,** has been added to the **PPCPAYTRAN** and **PPCEXTDATA** table.



The state of a batch can be DNE, OPEN, SENT, PARTIALCLOSED and CLOSED, where:

DNE means the batch object does not exist.

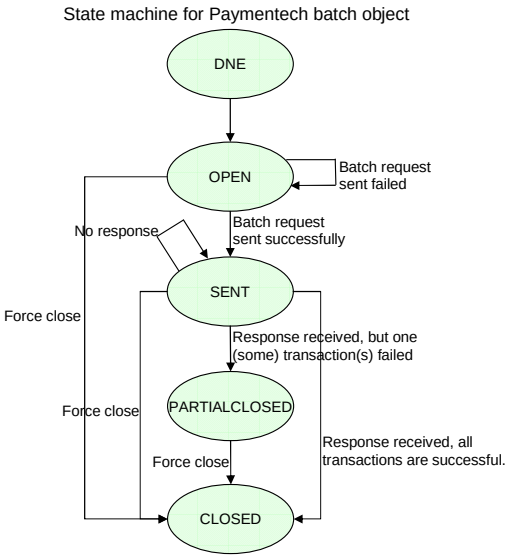
OPEN means that the batch object is created without being processed by the batch transaction scheduler;

SENT means that the batch object has been picked up by the batch transaction scheduler and the corresponding batch request has been sent to the Payment Service Provider gateway. No response has been received yet.

PARTIALCLOSED means that the response of the batch transaction has been received, but there are some payment transactions that have not been processed successfully by the Payment Service Provider.

CLOSED means that the response of the batch transaction has been received and all payment transactions in the batch have been successfully processed by the Payment Service Provider.

Paymentech Plug-in Controller enhancements (cont.)



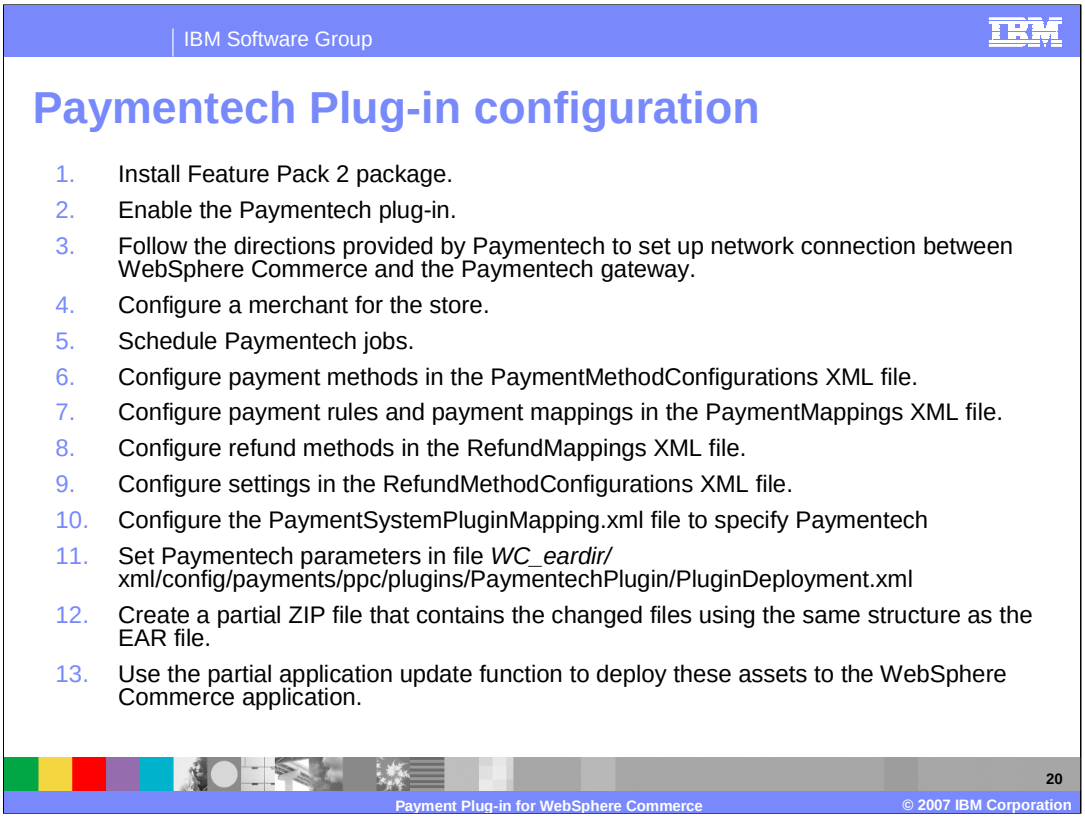
A tickler is generated when:

- The batch is not CLOSED
- The batch is not sent to the gateway
- getBatch failed to receive RFR response
- The batch becomes PARTIALCLOSED.

This diagram displays the state machine for the Paymentech batch object. After a batch object is created, its state is OPEN. The state is changed to SENT after the batch request has been sent successfully. If the received response shows that all transactions are successful, this batch object is CLOSED; if the received response shows that only some of the batch transactions are successful, the batch object state is PARTIALCLOSED.

You can see that if sending a batch object failed, the batch state stays at OPEN, and the Paymentech Plug-in Controller sends it again later. After retrying, if it still failed, its state remains OPEN until a payment administrator changes its state. The same theory applies for sending the batch object. If reading the response failed after retrying, the batch object remains SENT until a payment administrator changes its state. The maximum number of attempts to send and read a batch is configurable. This parameter is defined in the PluginDeployment.xml file for the Paymentech Plug-in. A payment administrator can force-close the batch using the **UI** after the transactions in the batch have been processed offline.

A tickler is generated if the following situations happen: 1. The batch cannot successfully become CLOSED; 2. The batch request cannot be sent to the Paymentech batch gateway; 3. The getBatch failed to receive RFR response from the Paymentech batch gateway; 4. The batch becomes PARTIALCLOSED.



Here are the configuration steps for using the Paymentech plug-in:

You need to install feature pack 2, then run the ant script to enable the Paymentech plug-in.

You need to follow directions provided by Paymentech to set up network connection between WebSphere Commerce and the Paymentech gateway. If you are not a Paymentech customer, you need to contact Paymentech to apply for an account.

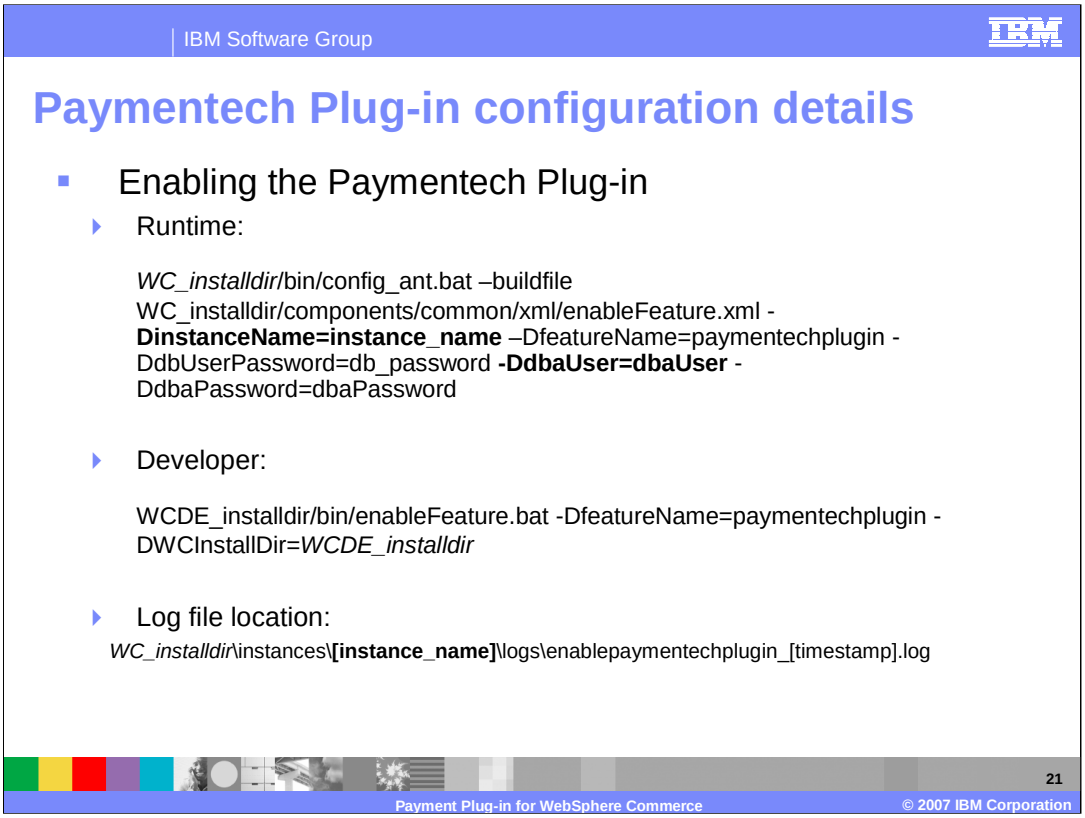
You need to configure a merchant for the store; Schedule Paymentech jobs for batch processing; Configure payment methods; Configure payment rules and payment mappings;

You need to Configure refund methods; Configure settings in the RefundMethodConfigurations XML file.

You need to specify Paymentech in PaymentSystemPluginMapping.xml file and set several Paymentech parameters in PluginDeployment.xml

Finally you need create a partial .zip file contains all the changes and deploy it into WebSphere Commerce application.

Step 6 – 10 and step 12, 13 are standard steps for all Payment plug-ins. The following slides discuss only steps only specific to the Paymentech plug-in.

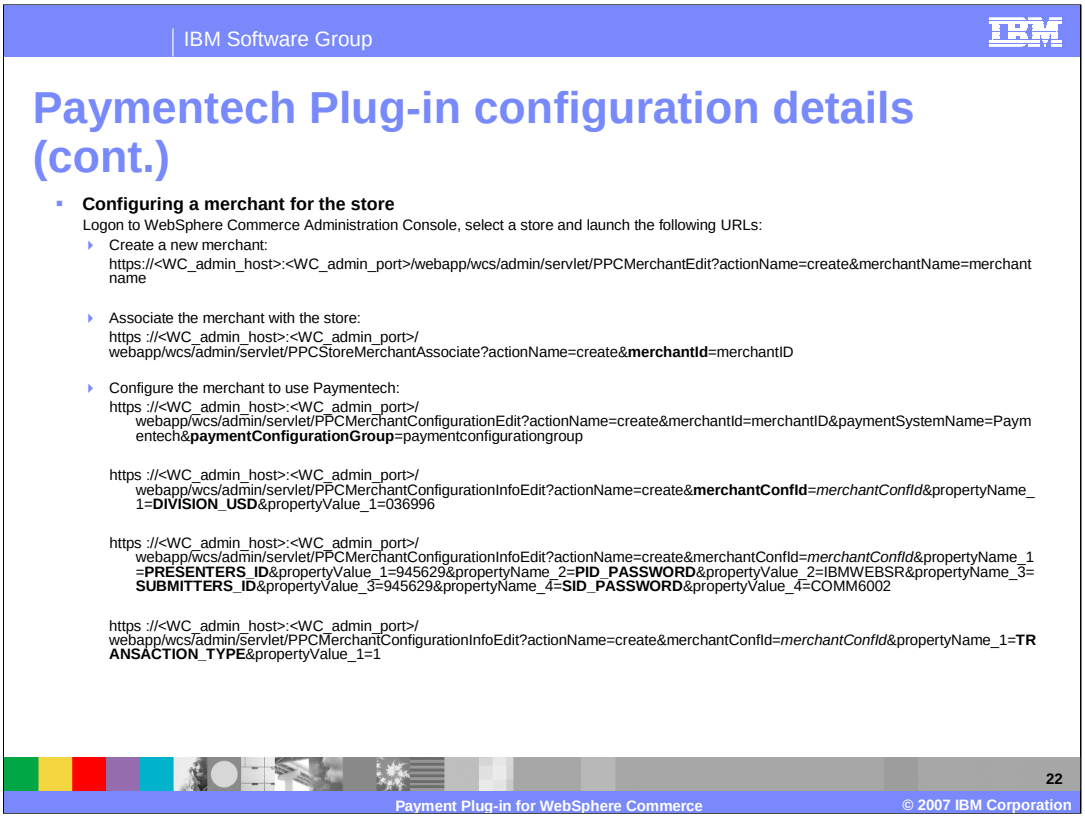


After installing feature pack 2, run the ant script to enable the Paymentech plug-in.

In the **runtime**, dbaUser and DdbaPassword(database administrator user ID and password) are required to update the database schema.

In the **developer** environment, DB user and password are not required because the WebSphere Commerce Developer uses Cloudscape as a database.

The log file for this script is under the WebSphere Commerce instance log directory.



The UI for configuring a merchant for a store is not yet available. To configure a merchant for a store, you need the WebSphere Commerce administration console's browser to launch the commands. You first need to login to the administration console, and select a store. Then place the URLs shown in this page into the URL field of the administration console browser. If the command runs successfully. you will see a blank page, otherwise you will receive an error page.

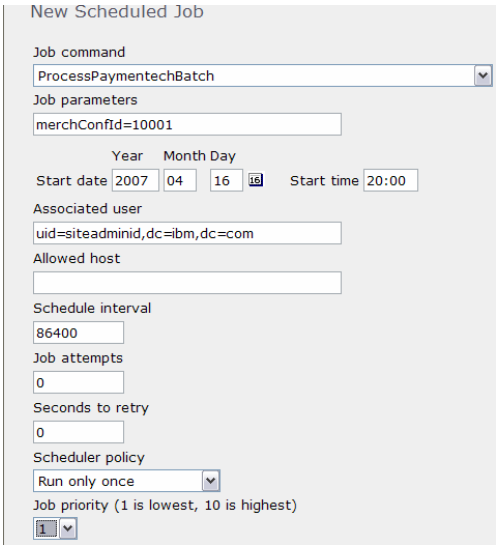
Associating the merchant with the store contains the merchantID parameter which can be found in the database table **MERCHANT.**

Configuring the merchant to use the Paymentech plug-in requires launching the URLs indicated with the following parameters:

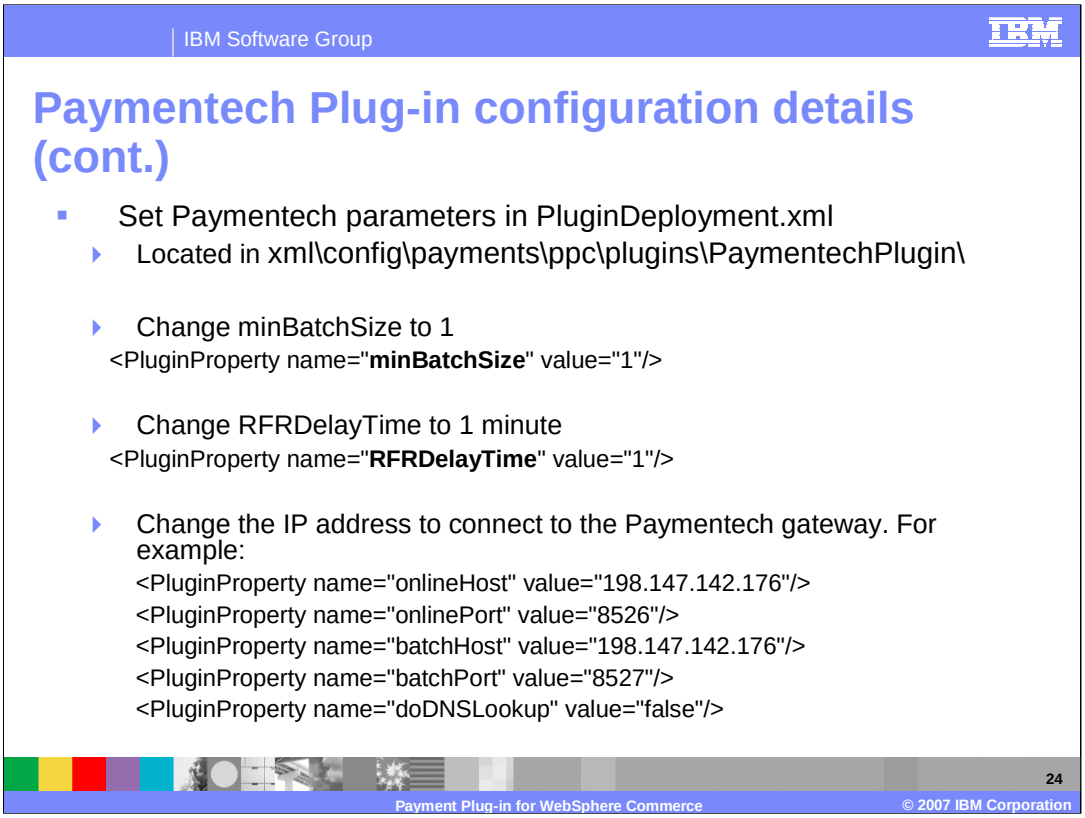
1. The first URL contains **paymentconfigurationgroup** which is the default if you did not set up a payment configuration group.
2. The second URL contains **merchantConfId** which can be found in the **merchconf** database table and **DIVISION_USD** which is the currency format found in the **MERCHCONFINFO.DIVISION_currencyCode** database table.
3. In the third and fourth URL, **PRESENTERS_ID, PID_PASSWORD, SUBMITTERS_ID, SID_PASSWORD, TRANSACTION_TYPE** are provided by Paymentech.

Paymentech Plug-in configuration details (cont.)

- Scheduling Paymentech jobs
  - In the Administration console, schedule ProcessPaymentechBatch job
    - Job parameters: merchConfid=xxx
    - Schedule interval=86400
    - Job attempts: 0
    - Seconds to retry: 0
    - Job priority: 1



In the Administration console, the ProcessPaymentechBatch job needs to be scheduled and the Schedule interval needs to be set to 86400 seconds which is equivalent to 24 hours.



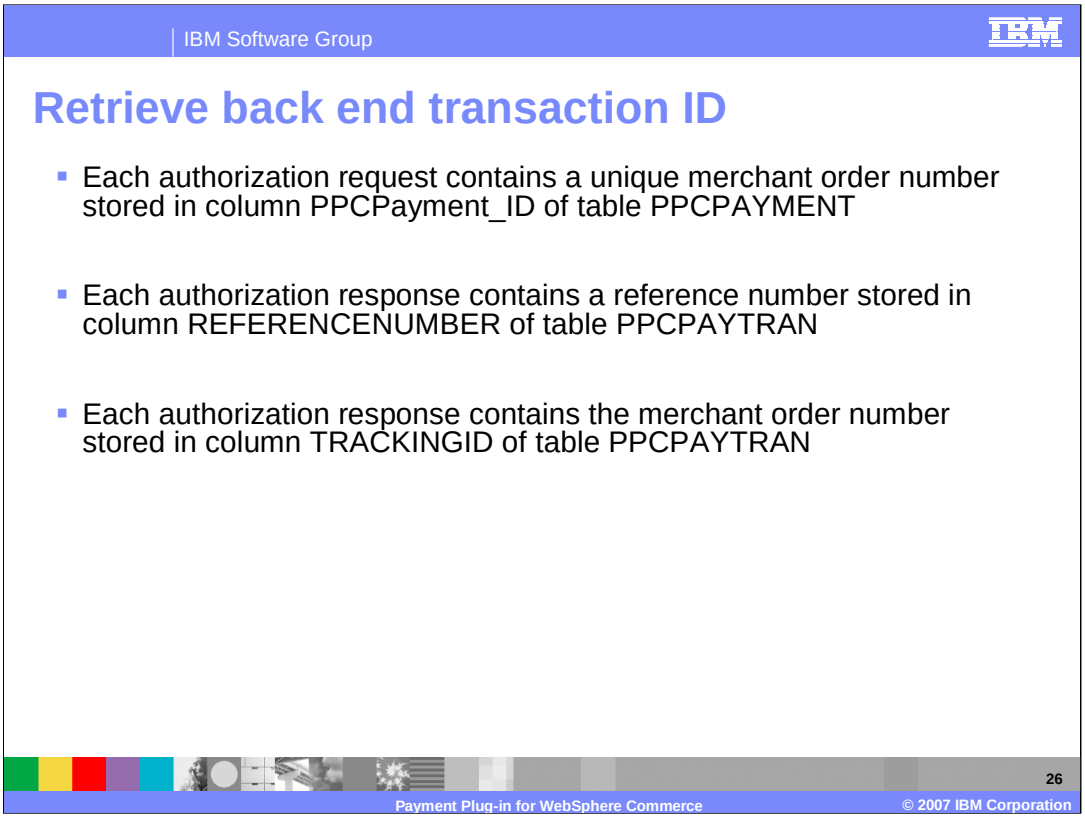
Each WebSphere Commerce payment plug-in contains a payment plug-in descriptor file called PluginDeployment.xml. This file is created when you run the ANT script to enable the Paymentech plug-in. Within this file, you need to change the parameters **minBatchSize** and **RFRDelayTime**. **minBatchSize** refers to the minimum number of transactions in one batch. If the number of deposit and credit transactions is less than the minBatchSize, then nothing is done. **RFRDelayTime** refers to the delay time between the time the batch request was sent to Paymentech and the time the Request For Response was sent to Paymentech to retrieve the batch response.

In this file, you also need to change the value of the properties for the Paymentech online gateway and batch gateway, such as **onlineHost**, **onlinePort**, **batchHost** and **batchPort**. This information can be obtained from Paymentech.

Section

Paymentech Plug-in problem determination

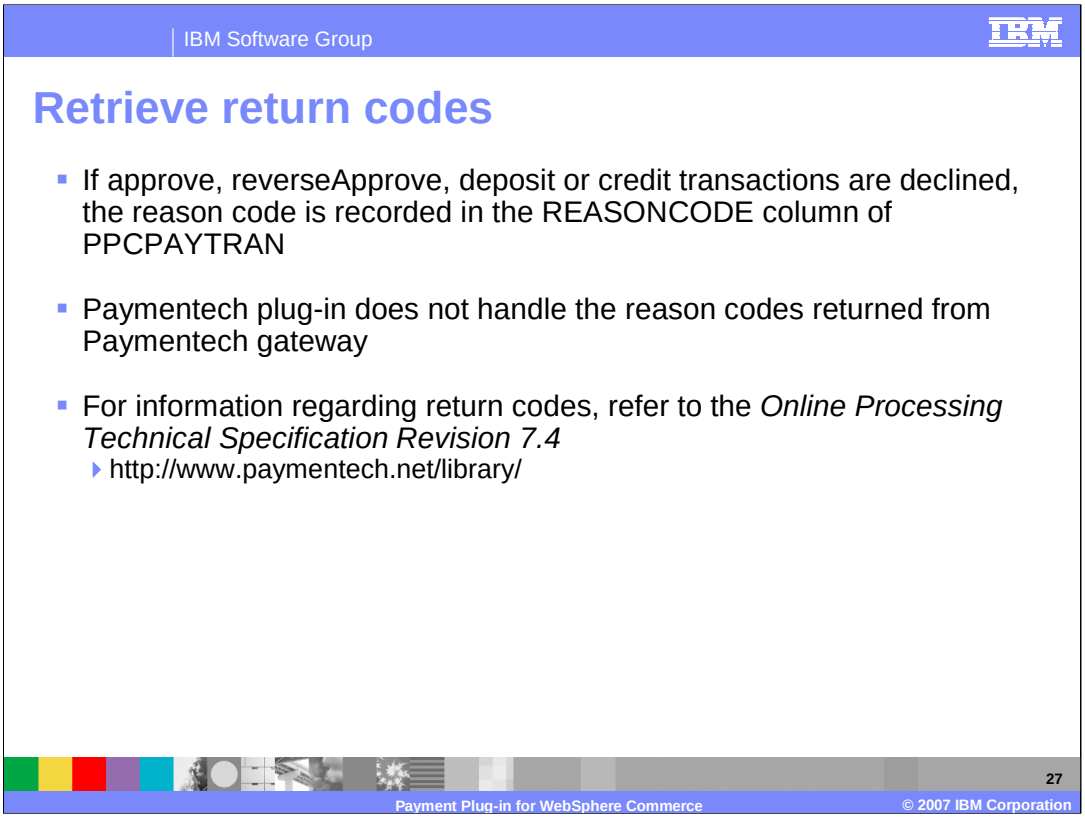
This section discusses Paymentech Plug-in problem determination.



Paymentech requires a unique identifier per authorization transaction and in the Paymentech plug-in the **merchant order number** is used for this purpose. It is saved in column PPCPayment_ID of the PPCPAYMENT table.

Keep in mind that the **merchant order number** is not the order number. On the WebSphere Commerce side, due to multiple payment methods support, the order number in WebSphere Commerce cannot be used as the merchant order number since there may be multiple payment authorization transactions per order. Instead, the PPCPayment_ID is adopted since the payment authorization transaction has a one-to-one correlation to the payment object in the Paymentech Plug-in Controller.

In the Paymentech authorization response, the merchant order number is sent back with a Paymentech reference number. This reference number is stored in the column REFERENCENUMBER of the PPCPAYTRAN table and the merchant order number is stored in the column TRACKINGID of the PPCPAYTRAN table.



If approve, reverseApprove, deposit or credit transactions in the batch are declined by the Paymentech gateway, Paymentech sends back the reason codes in the response messages explaining why the transactions were declined. The reason code is stored in the REASONCODE column of the PPCPAYTRAN table.

Note that the Paymentech plug-in does not handle the reason codes returned from the Paymentech gateway.

If you want to know what the reason codes mean, visit www.paymentech.net/library/

Recover from transaction failure

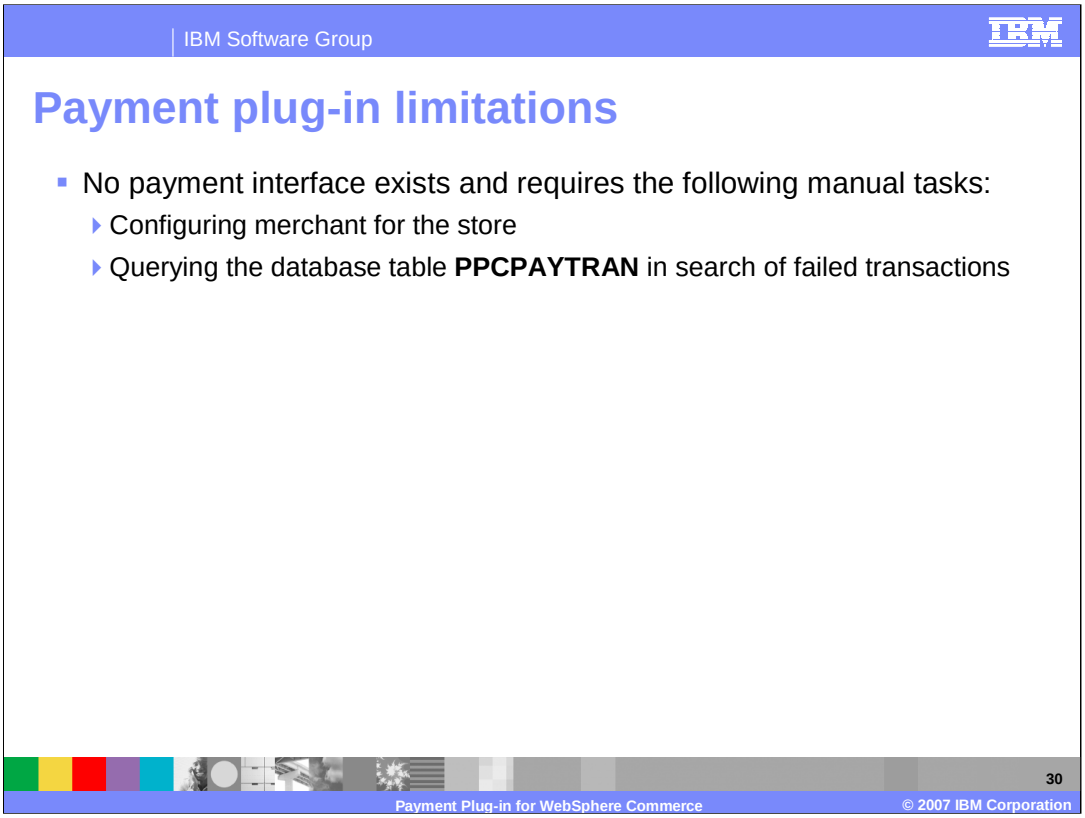
- Re-send the batch, re-send the authentication and re-read the response are used as internal retry logic
- Retry functionality is the only recovery mechanism
- Payment administrators have to manually deal with failed transactions

The Paymentech Plug-in module has internal retry logic, such as re-send the batch, re-send the authentication and re-read the response. The maximum number of re-send or re-read is configurable, however, there is no special recovery mechanism besides the retry function. Payment administrators have to manually deal with the failed transactions.

Section

Known limitations

This section discusses the known limitations within the Paymentech Plug-in.



Currently, there is no payment interface for feature pack 2 and you have to manually accomplish the following tasks:

1. Configure a merchant store using the administration console URL field.
2. Manually query the database to retrieve the transaction data.

Summary

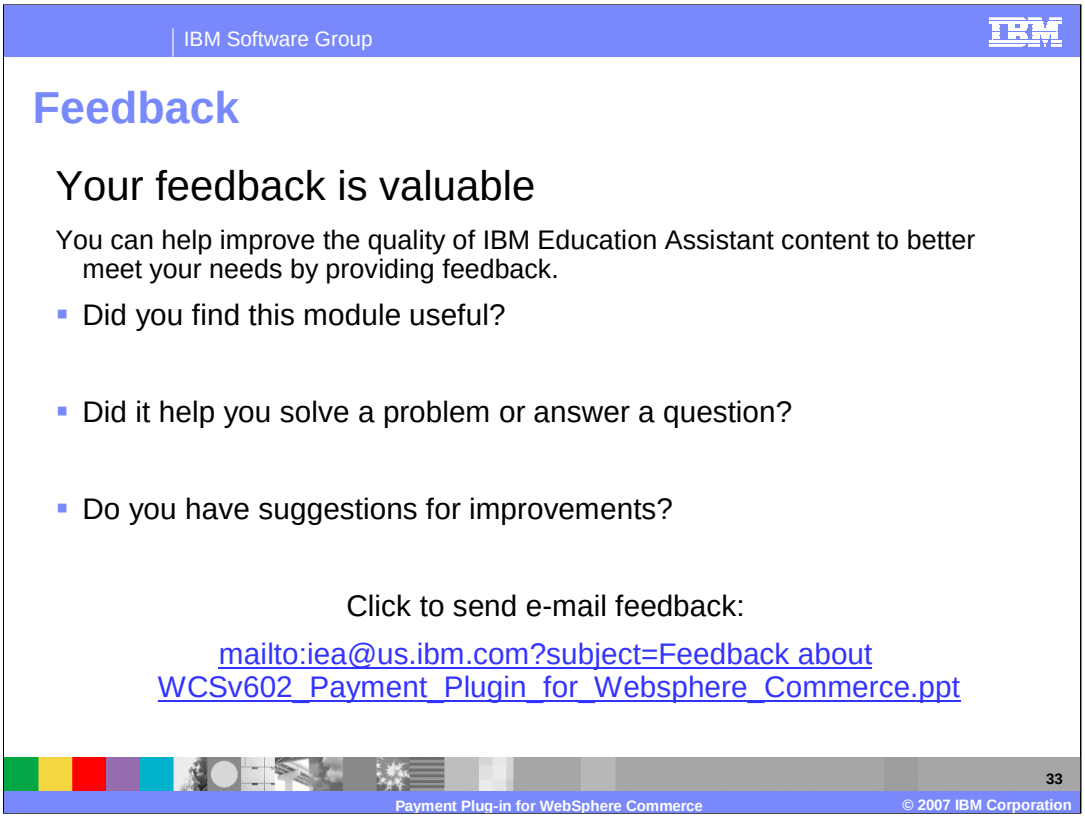
- Reviewed WebSphere Commerce V6.0 payment system
- Introduced Paymentech Plug-in implementation details
- Discussed Paymentech Plug-in problem determination
- Talked some of known limitations.

In summary, this presentation discussed the WebSphere Commerce V6.0 payment system, introduced Paymentech Plug-in implementation details, discussed Paymentech Plug-in problem determination and disclosed some of the known limitations.

References

- Paymentech technical specifications
  http://www.paymentech.net/library/
- Tutorial: Developing a payment plug-in for WebSphere Commerce Payments
  http://publib.boulder.ibm.com/infocenter/wchelp/v6r0m0/topic/com.ibm.commerce.payments.events.doc/tutorial/tpp1.htm

For more information regarding Paymentech technical specifications or for a tutorial that will assist you in developing your own payment plug-in, visit the sites indicated in the slide.



You can help improve the quality of IBM Education Assistant content by providing feedback.

Template Revision: 04/25/2006 11:09 AM

Trademarks, copyrights, and disclaimers

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

IBM	CICS	IMS	MQSeries	Tivoli
IBM(logo)	Cloudscape	Informix	OS/390	WebSphere
e(logo)business	DB2	iSeries	OS/400	xSeries
AIX	DB2 Universal Database	Lotus	pSeries	zSeries

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

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

Intel, ActionMedia, LANDesk, MMX, Pentium and ProShare are trademarks of Intel Corporation in the United States, other countries, or both.

UNIX is a registered trademark of The Open Group in the United States and other countries.

Linux is a registered trademark of Linus Torvalds.

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

Product data has been reviewed for accuracy as of the date of initial publication. Product data is subject to change without notice. This document could include technical inaccuracies or typographical errors. IBM may make improvements and/or changes in the product(s) and/or program(s) described herein at any time without notice. Any statements regarding IBM's future direction and intent are subject to change or withdrawal without notice, and represent goals and objectives only. References in this document to IBM products, programs, or services does not imply that IBM intends to make such products, programs or services available in all countries in which IBM operates or does business. Any reference to an IBM Program Product in this document is not intended to state or imply that only that program product may be used. Any functionally equivalent program, that does not infringe IBM's intellectual property rights, may be used instead.

Information is provided "AS IS" without warranty of any kind. THE INFORMATION PROVIDED IN THIS DOCUMENT IS DISTRIBUTED "AS IS" WITHOUT ANY WARRANTY, EITHER EXPRESS OR IMPLIED. IBM EXPRESSLY DISCLAIMS ANY WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE OR NONINFRINGEMENT. IBM shall have no responsibility to update this information. IBM products are warranted, if at all, according to the terms and conditions of the agreements (e.g., IBM Customer Agreement, Statement of Limited Warranty, International Program License Agreement, etc.) under which they are provided. Information concerning non-IBM products was obtained from the suppliers of those products, their published announcements or other publicly available sources. IBM has not tested those products in connection with this publication and cannot confirm the accuracy of performance, compatibility or any other claims related to non-IBM products. IBM makes no representations or warranties, express or implied, regarding non-IBM products and services.

The provision of the information contained herein is not intended to, and does not, grant any right or license under any IBM patents or copyrights. Inquiries regarding patent or copyright licenses should be made, in writing, to:

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

Performance is based on measurements and projections using standard IBM benchmarks in a controlled environment. All customer examples described are presented as illustrations of how those customers have used IBM products and the results they may have achieved. The actual throughput or performance that any user will experience will vary depending upon considerations such as the amount of multiprogramming in the user's job stream, the I/O configuration, the storage configuration, and the workload processed. Therefore, no assurance can be given that an individual user will achieve throughput or performance improvements equivalent to the ratios stated here.

Note to U.S. Government Users - Documentation related to restricted rights-Use, duplication or disclosure is subject to restrictions set forth in GSA ADP Schedule Contract and IBM Corp.