IBM Software Group

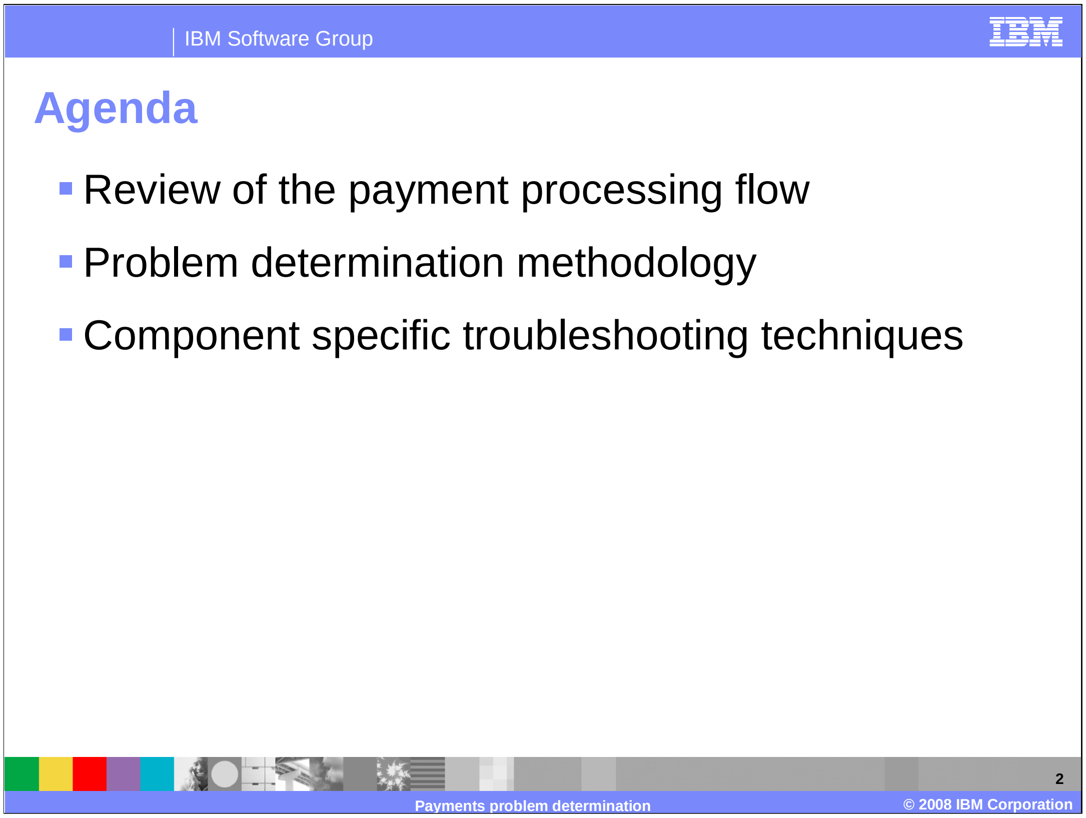
WebSphere® Commerce V6.0

Payments problem determination

Updated May 7, 2008

This presentation covers the WebSphere Commerce version 6 payments problem determination presentation.

This presentation will provide an in-depth look at how to troubleshoot problems with the WebSphere Commerce Payments component



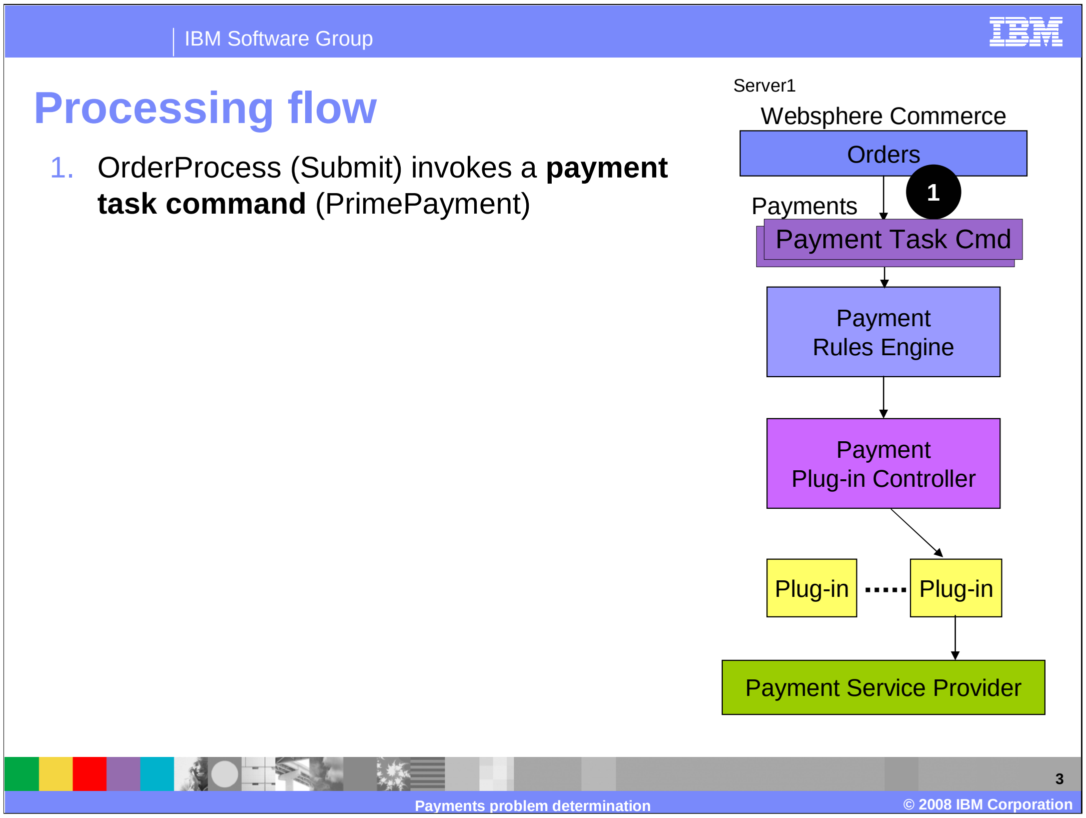
Agenda

- Review of the payment processing flow
- Problem determination methodology
- Component specific troubleshooting techniques

In this presentation, these topics are discussed:

The payment processing flow is reviewed. Next the general problem determination methodology is discussed.

More specific sub-component details are explored. Specific techniques are discussed that can be used to identify problems and troubleshoot in each sub-component.



IBM Software Group

Processing flow

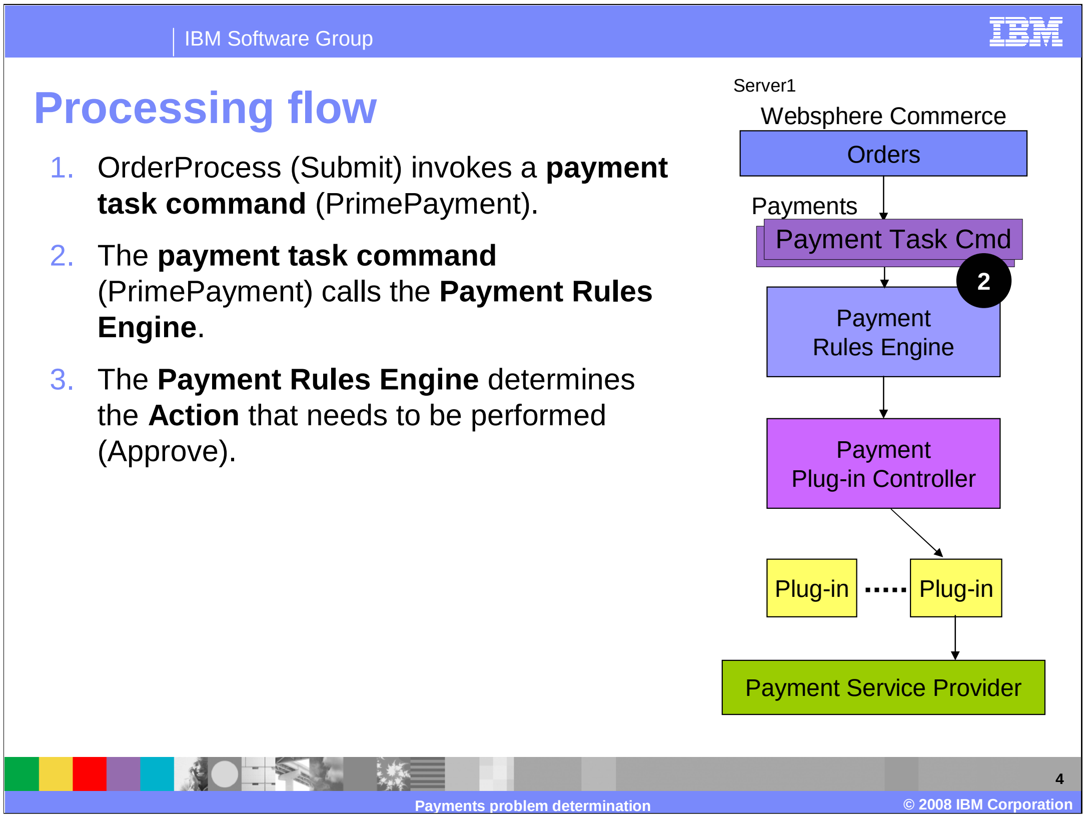
1. OrderProcess (Submit) invokes a **payment task command** (PrimePayment)

3

Payments problem determination

This section reviews the payment flow.

The payment flow starts when a WebSphere Commerce Controller command is invoked that requires some type of payment processing. For example, clicking submit on the shopping cart page calls the OrderProcess command. This in turn invokes the payments sub-system by calling the PrimePayment task command.



IBM Software Group

Processing flow

1. OrderProcess (Submit) invokes a **payment task command** (PrimePayment).
2. The **payment task command** (PrimePayment) calls the **Payment Rules Engine**.
3. The **Payment Rules Engine** determines the **Action** that needs to be performed (Approve).

Server1

Websphere Commerce

Orders

Payments

Payment Task Cmd

2

Payment Rules Engine

Payment Plug-in Controller

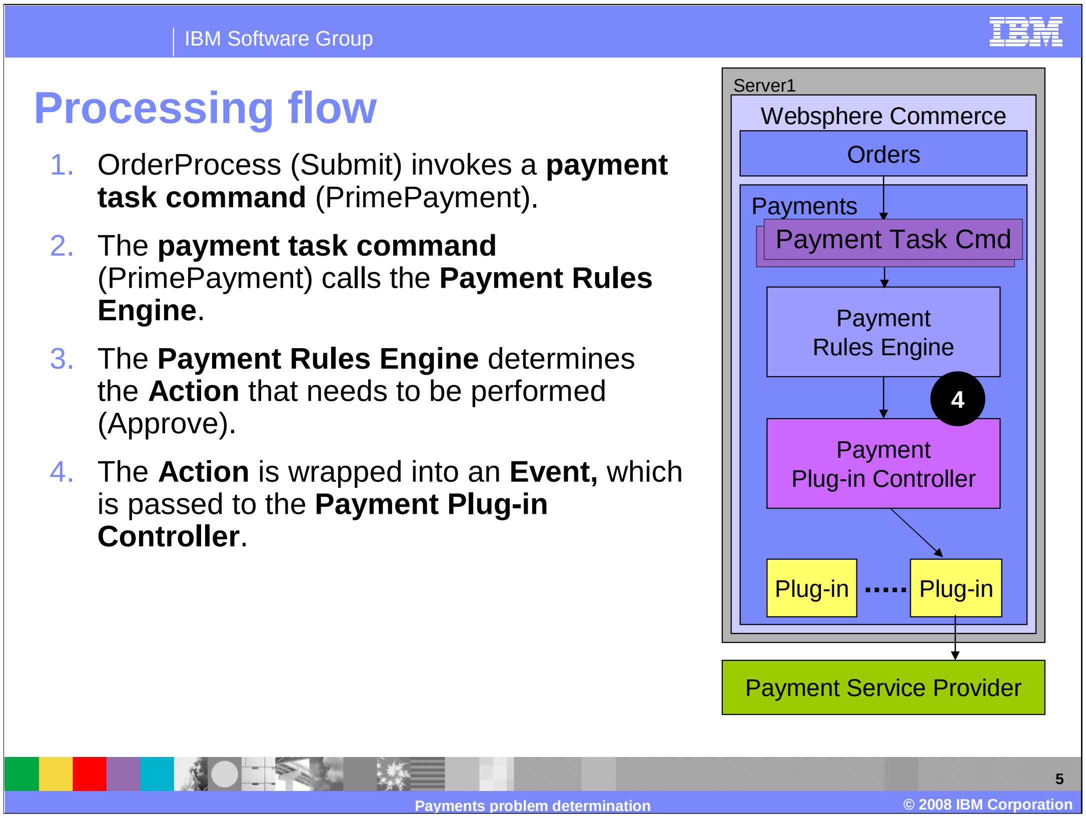
Plug-in ····· Plug-in

Payment Service Provider

4

Payments problem determination

In step 2 and 3, the payment policy command calls the Payment Rules Engine component which determines which action needs to be performed.



IBM Software Group

Processing flow

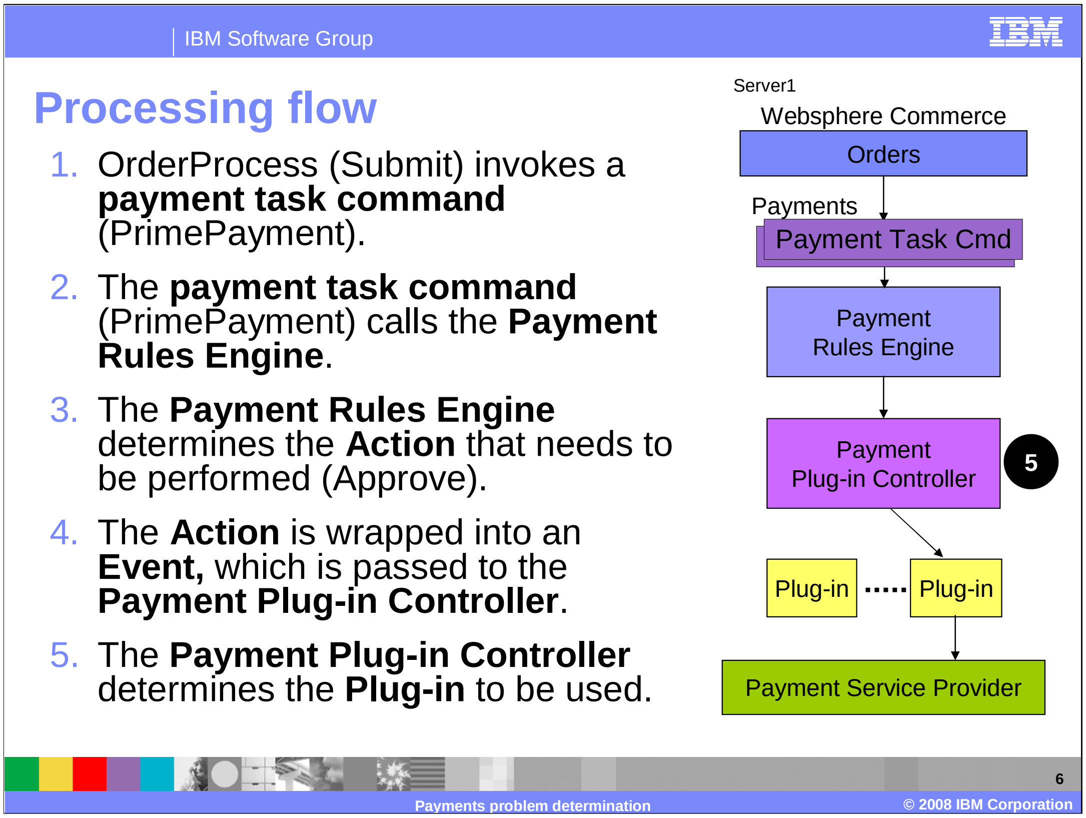
1. OrderProcess (Submit) invokes a **payment task command** (PrimePayment).
2. The **payment task command** (PrimePayment) calls the **Payment Rules Engine**.
3. The **Payment Rules Engine** determines the **Action** that needs to be performed (Approve).
4. The **Action** is wrapped into an **Event,** which is passed to the **Payment Plug-in Controller**.

5

Payments problem determination

© 2008 IBM Corporation

In step 4, the action is going to be wrapped into an event, which is passed to the Payment Plug-in Controller.



IBM Software Group

Processing flow

1. OrderProcess (Submit) invokes a **payment task command** (PrimePayment).
2. The **payment task command** (PrimePayment) calls the **Payment Rules Engine**.
3. The **Payment Rules Engine** determines the **Action** that needs to be performed (Approve).
4. The **Action** is wrapped into an **Event,** which is passed to the **Payment Plug-in Controller**.
5. The **Payment Plug-in Controller** determines the **Plug-in** to be used.

Server1

Websphere Commerce

Orders

Payments

Payment Task Cmd

Payment Rules Engine

Payment Plug-in Controller

5

Plug-in ····· Plug-in

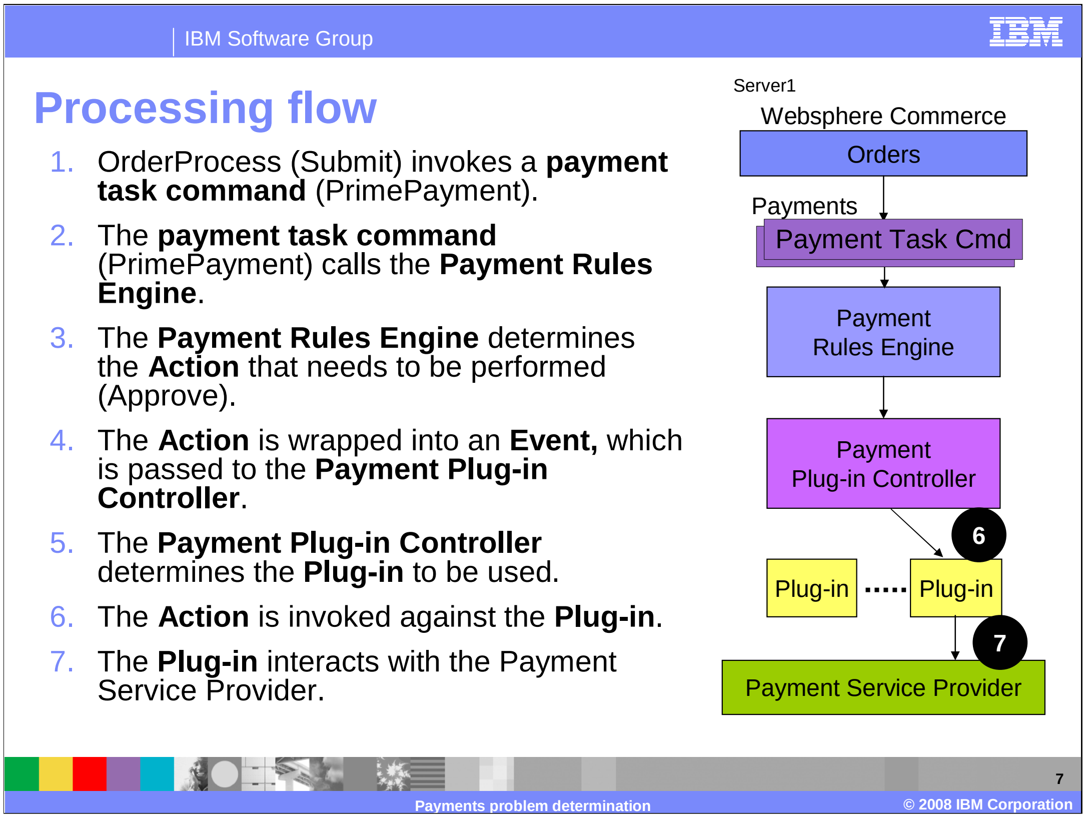
Payment Service Provider

6

Payments problem determination

© 2008 IBM Corporation

In step 5, the payment plug-in controller determines which plug-in services the given request.



IBM Software Group

Processing flow

1. OrderProcess (Submit) invokes a **payment task command** (PrimePayment).
2. The **payment task command** (PrimePayment) calls the **Payment Rules Engine**.
3. The **Payment Rules Engine** determines the **Action** that needs to be performed (Approve).
4. The **Action** is wrapped into an **Event,** which is passed to the **Payment Plug-in Controller**.
5. The **Payment Plug-in Controller** determines the **Plug-in** to be used.
6. The **Action** is invoked against the **Plug-in**.
7. The **Plug-in** interacts with the Payment Service Provider.

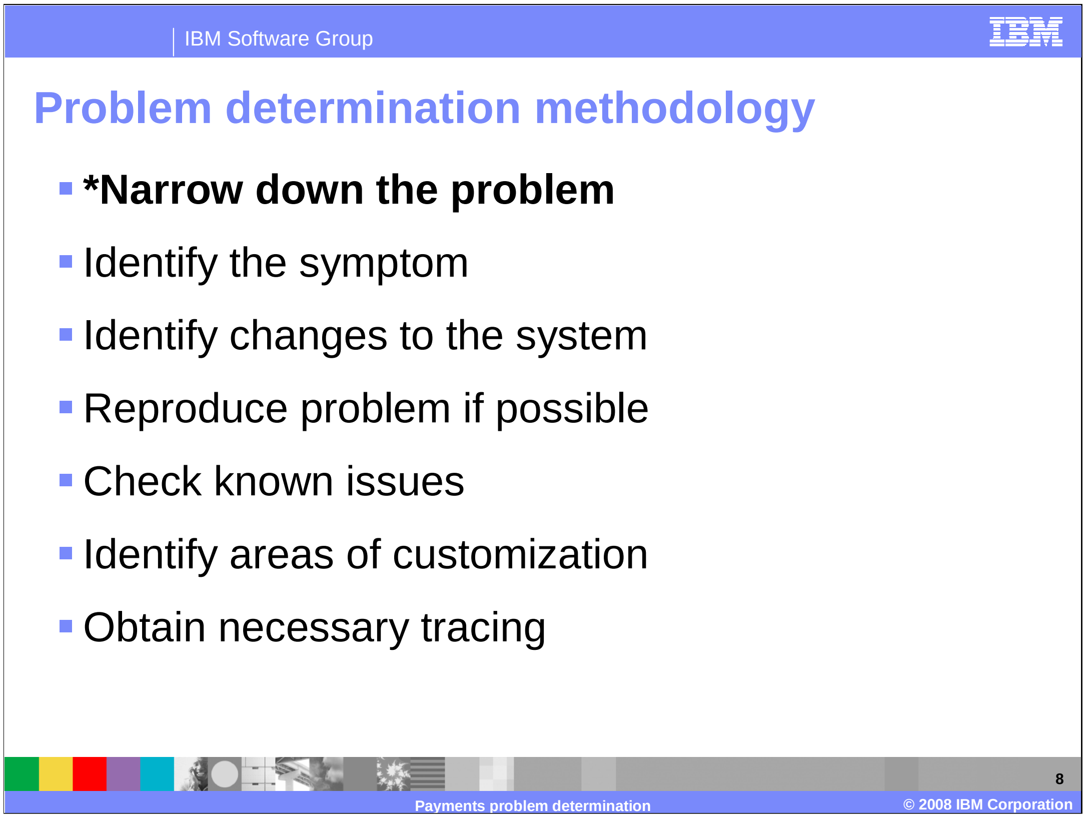
7

Payments problem determination

© 2008 IBM Corporation

In step 6, the payment plug-in controller invokes the action against the plug-in, and the plug-in communicates with the backend payment service provider in step 7.

If you are interested in a more in-depth look at the payment processing flow, see the WebSphere Commerce version 6 payments technical overview presentation, which is also available in IBM® Education Assistant.



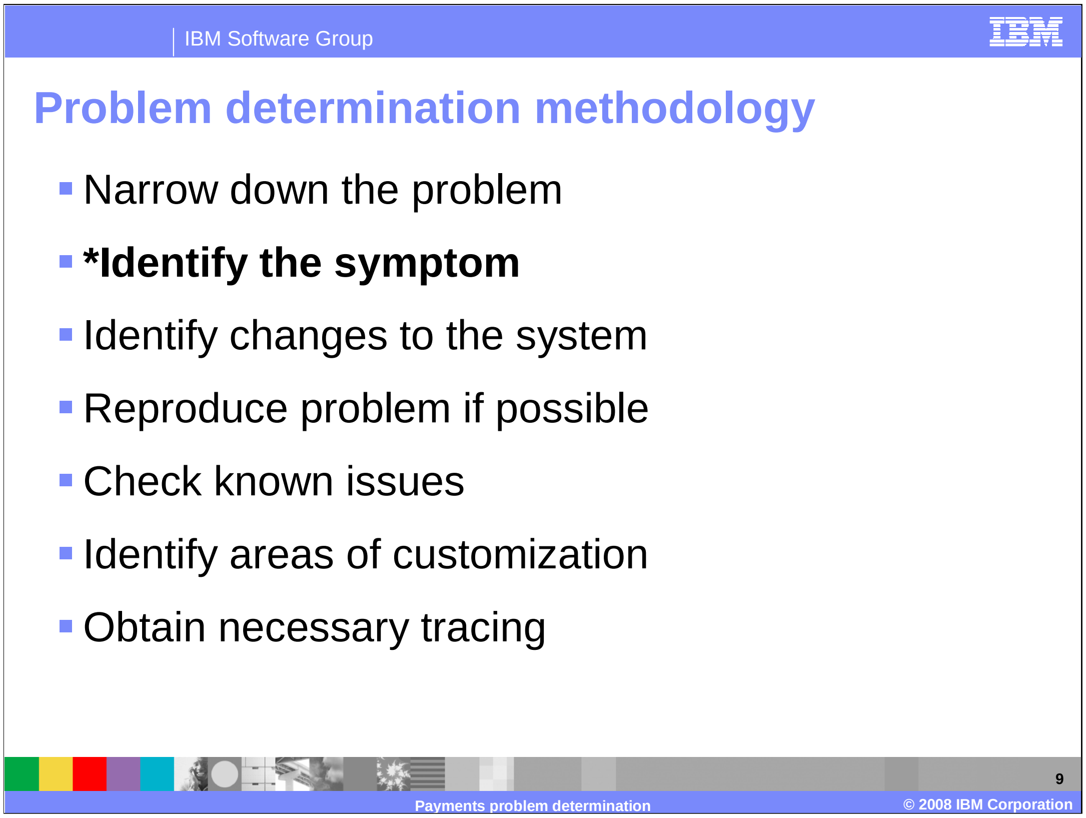
Problem determination methodology

- ***Narrow down the problem**
- Identify the symptom
- Identify changes to the system
- Reproduce problem if possible
- Check known issues
- Identify areas of customization
- Obtain necessary tracing

This section reviews the problem determination methodology.

The most important part of WebSphere Commerce payments troubleshooting is to narrow down the problem.

For example, the idea is to go from “I have a problem with submitting payments” to “I have a problem between the payment plug-in and the backend payment service provider”.

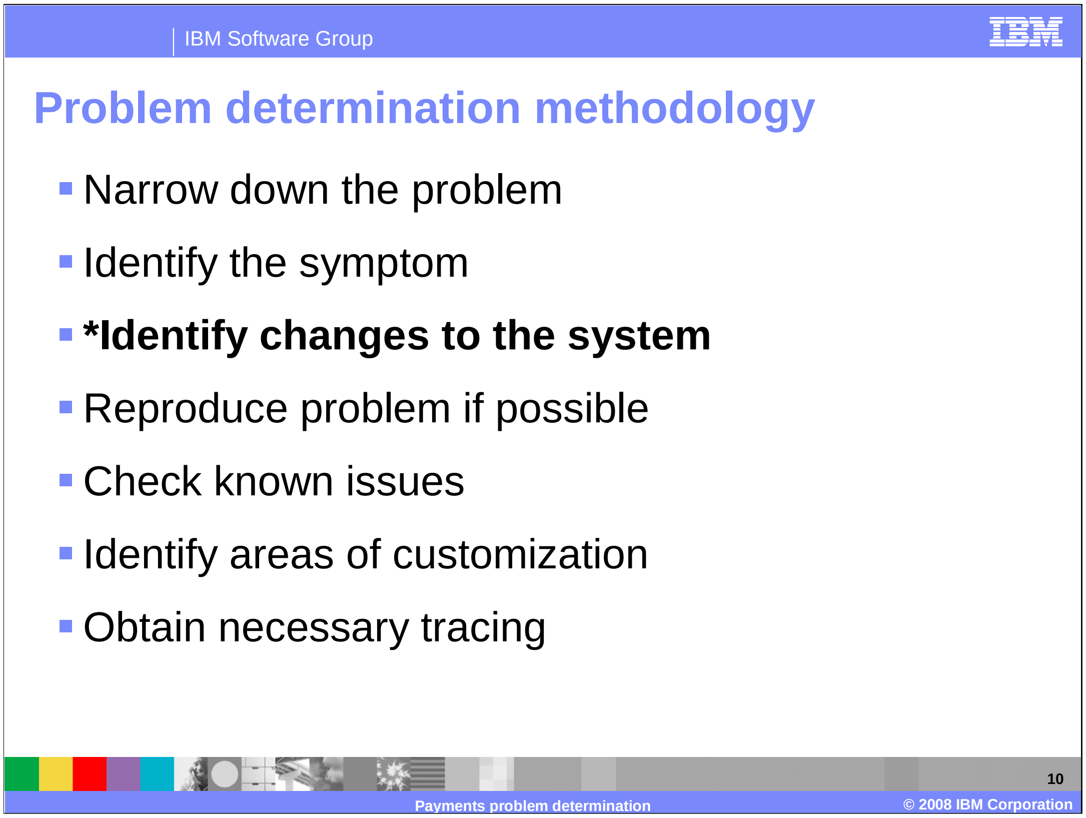


Problem determination methodology

- Narrow down the problem
- ***Identify the symptom**
- Identify changes to the system
- Reproduce problem if possible
- Check known issues
- Identify areas of customization
- Obtain necessary tracing

In order to help narrow down the problem, you can use this guideline:

The first step in troubleshooting is identifying the symptoms. A payment problem can happen in any step of the processing flow. Understanding the symptoms gives you a better idea of where to begin investigating. You might want to ask yourself questions such as: Was there an error message? What time did the error occur? Is the problem repeated continuously?



IBM Software Group

Problem determination methodology

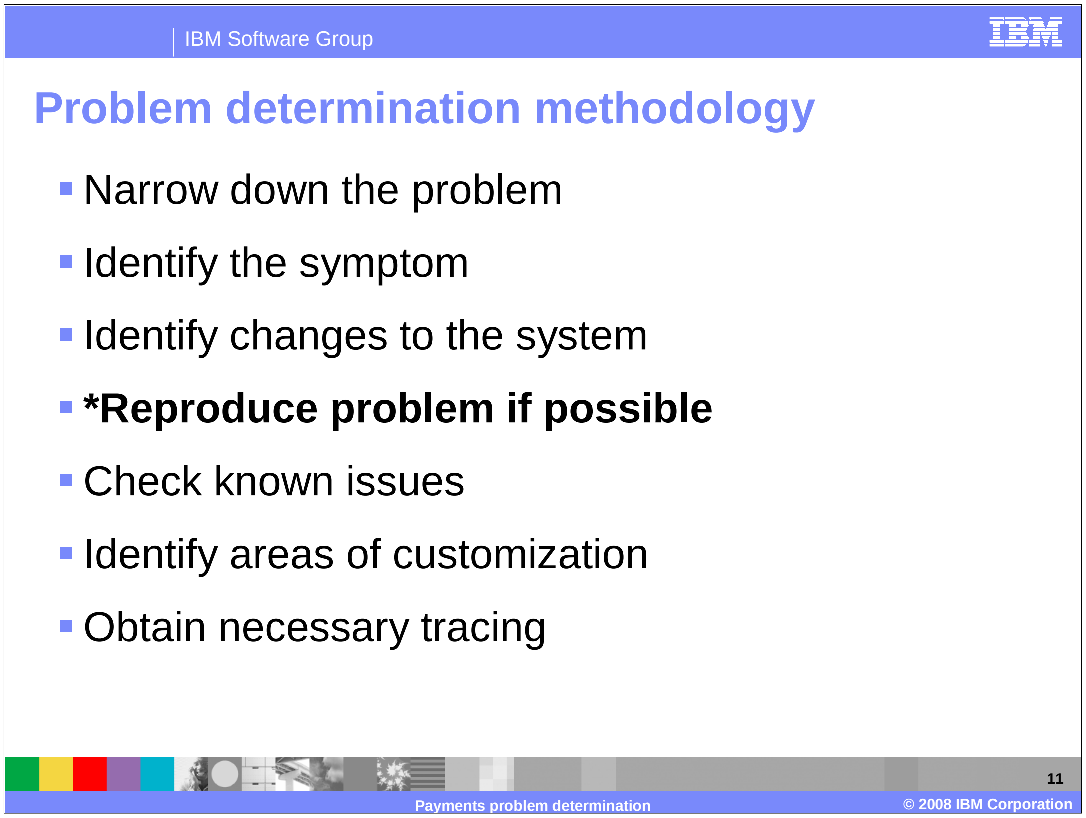
- Narrow down the problem
- Identify the symptom
- ***Identify changes to the system**
- Reproduce problem if possible
- Check known issues
- Identify areas of customization
- Obtain necessary tracing

10

Payments problem determination

© 2008 IBM Corporation

The next step is to look at whether anything changed on the system. That is, did the same scenario work previously? If so, what changed? If nothing was changed in your payments configuration, there might be a change in the payment service provider.



IBM Software Group

Problem determination methodology

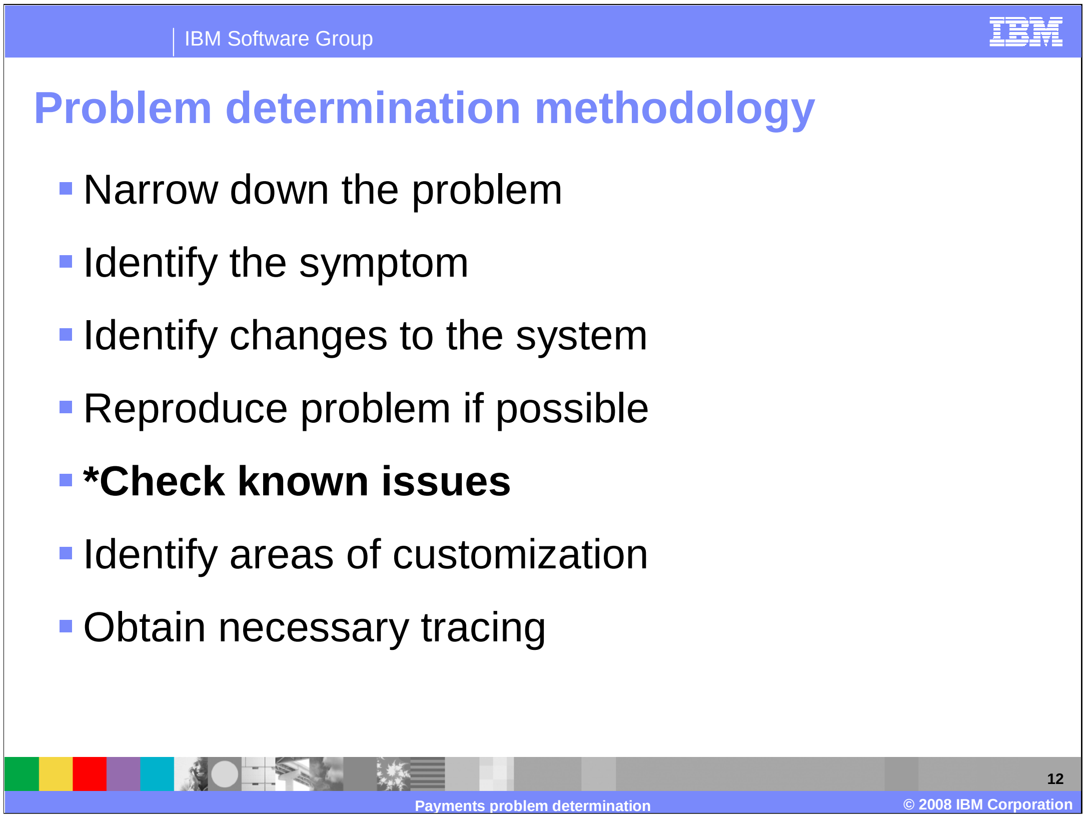
- Narrow down the problem
- Identify the symptom
- Identify changes to the system
- ***Reproduce problem if possible**
- Check known issues
- Identify areas of customization
- Obtain necessary tracing

11

Payments problem determination

© 2008 IBM Corporation

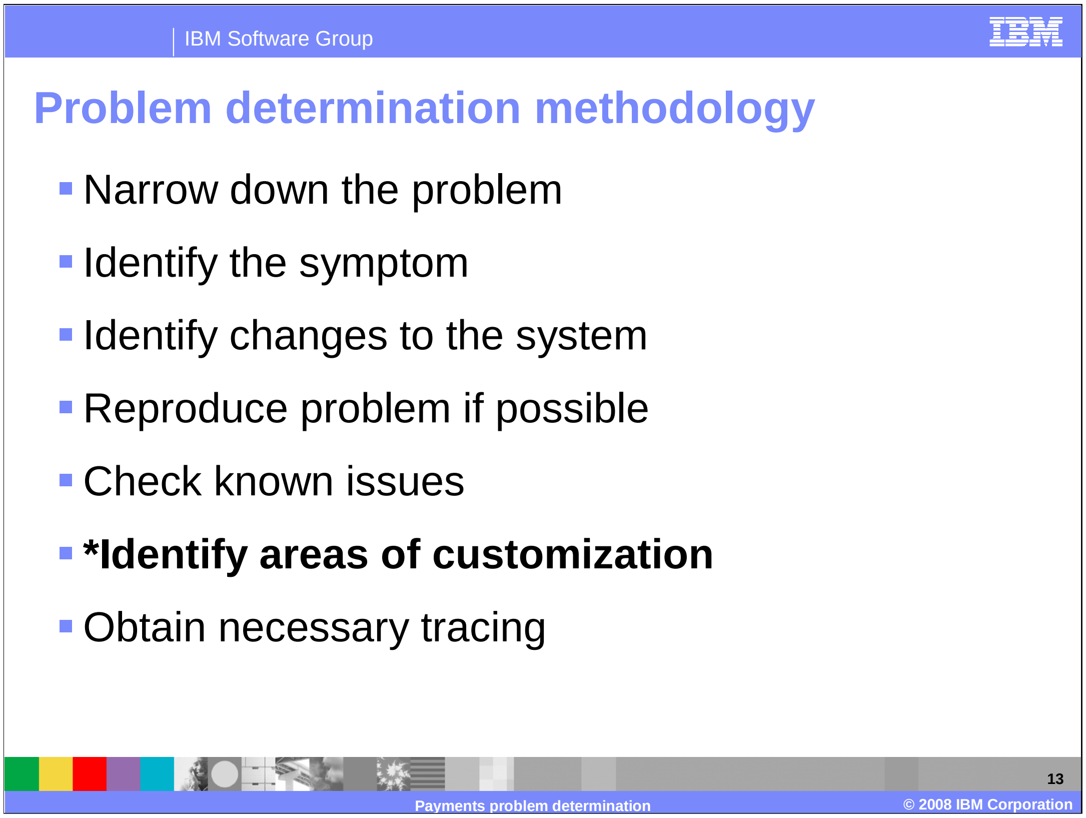
If possible, reproduce the problem. A problem that can be reproduced has a much better chance of being solved quickly.



Problem determination methodology

- Narrow down the problem
- Identify the symptom
- Identify changes to the system
- Reproduce problem if possible
- ***Check known issues**
- Identify areas of customization
- Obtain necessary tracing

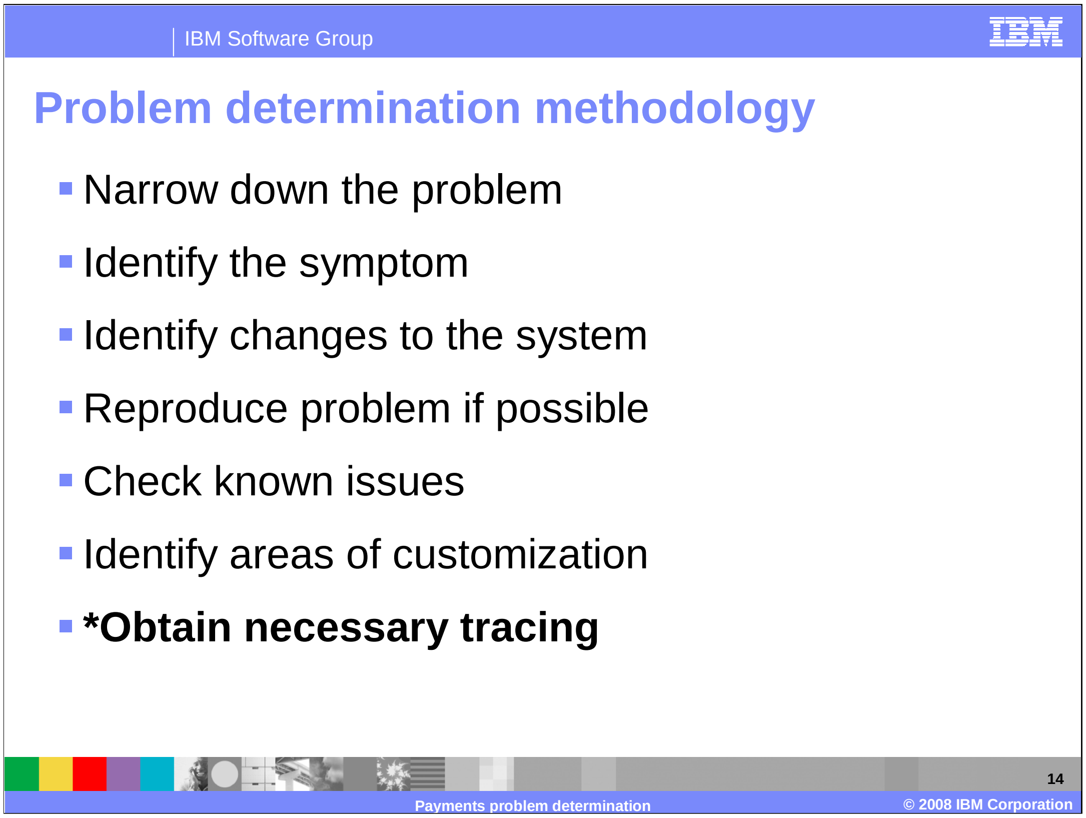
Next, it is a good idea to check the known issues that are published on the WebSphere Commerce support site. IBM publishes many technotes that provide solutions to known payments problems and can be a quick way to resolve the issues.



Problem determination methodology

- Narrow down the problem
- Identify the symptom
- Identify changes to the system
- Reproduce problem if possible
- Check known issues
- ***Identify areas of customization**
- Obtain necessary tracing

Identifying areas of customization is also important. Payment system is highly customizable because most users have a unique way of handling their payment processing. It is helpful to know where customizations have been added to better understand any differences in the processing flow.



IBM Software Group

Problem determination methodology

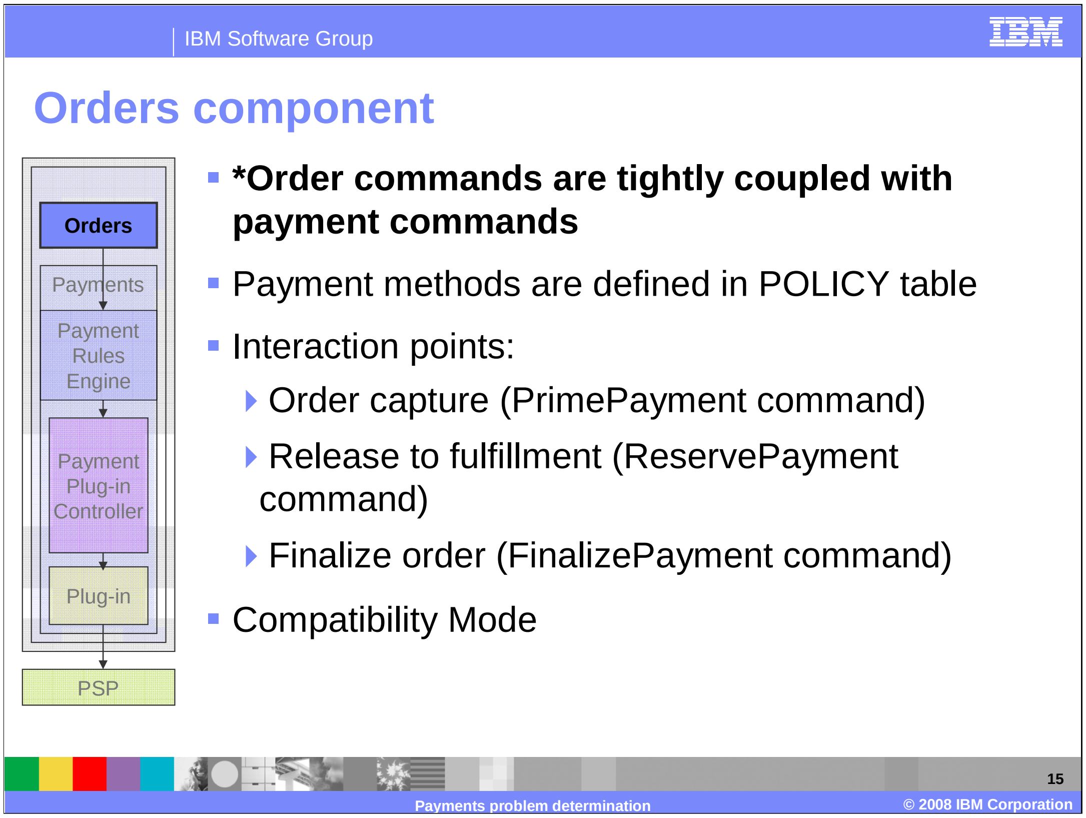
- Narrow down the problem
- Identify the symptom
- Identify changes to the system
- Reproduce problem if possible
- Check known issues
- Identify areas of customization
- ***Obtain necessary tracing**

14

Payments problem determination

Finally, if further investigation is required, gathering trace data helps pinpoint the source of the problem and is likely required if you need to contact IBM support.

The next slides cover the necessary tracing for each sub-component.



Orders component

Orders

Payments

Payment Rules Engine

Payment Plug-in Controller

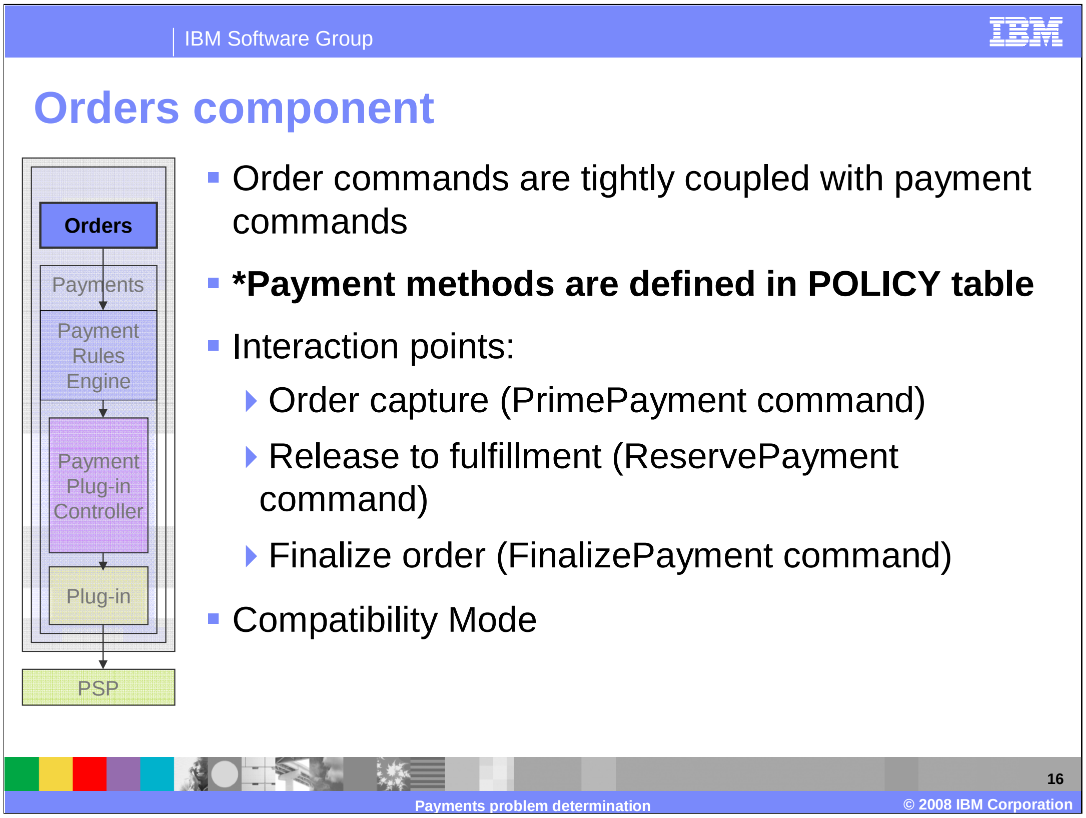
Plug-in

PSP

- ***Order commands are tightly coupled with payment commands**
- Payment methods are defined in POLICY table
- Interaction points:
  - Order capture (PrimePayment command)
  - Release to fulfillment (ReservePayment command)
  - Finalize order (FinalizePayment command)
- Compatibility Mode

Take a look each sub-component to see how to isolate problems in each area and to troubleshoot them if they arise.

First, examine the Orders component. The payments processing cycle begins with the Orders component. The order and payment components are tightly coupled. The key to debugging problems in this area is understanding where the interaction points are between orders and payments. That is, at what point in the order flow does an order command interact with the payment subsystem.


Orders
Payments
Payment Rules Engine
Payment Plug-in Controller
Plug-in
PSP

IBM Software Group

Orders component

- Order commands are tightly coupled with payment commands
- ***Payment methods are defined in POLICY table**
- Interaction points:
  - Order capture (PrimePayment command)
  - Release to fulfillment (ReservePayment command)
  - Finalize order (FinalizePayment command)
- Compatibility Mode

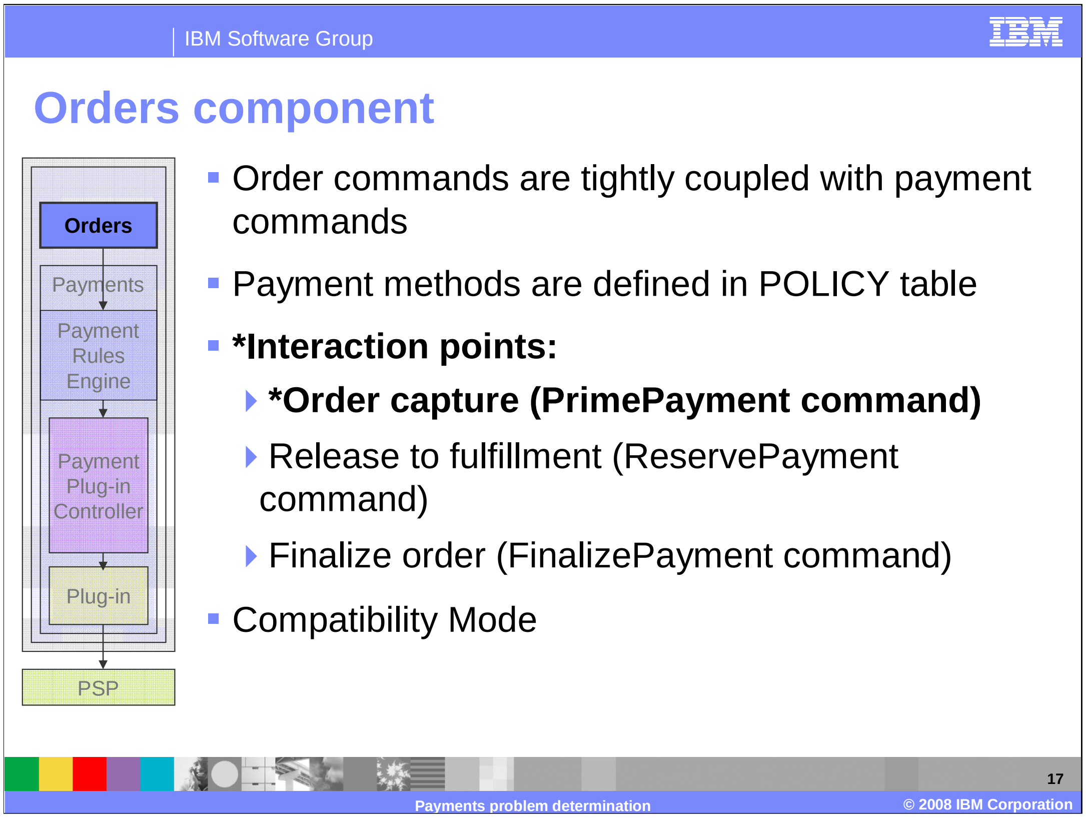
16

Payments problem determination

© 2008 IBM Corporation

The very first interaction between Orders and Payments happens even before the order is placed. In the checkout page, the shopper is presented a list of available payment methods, such as VISA or MasterCard. These methods are defined in the POLICY table.

When defining the POLICY table for your store, keep in mind the properties column which defines various characteristics of your payment method. Specifically, check if the POLICY table has the correct payment configuration group, whether it is enabled for the store, and whether it is being used in compatible mode.



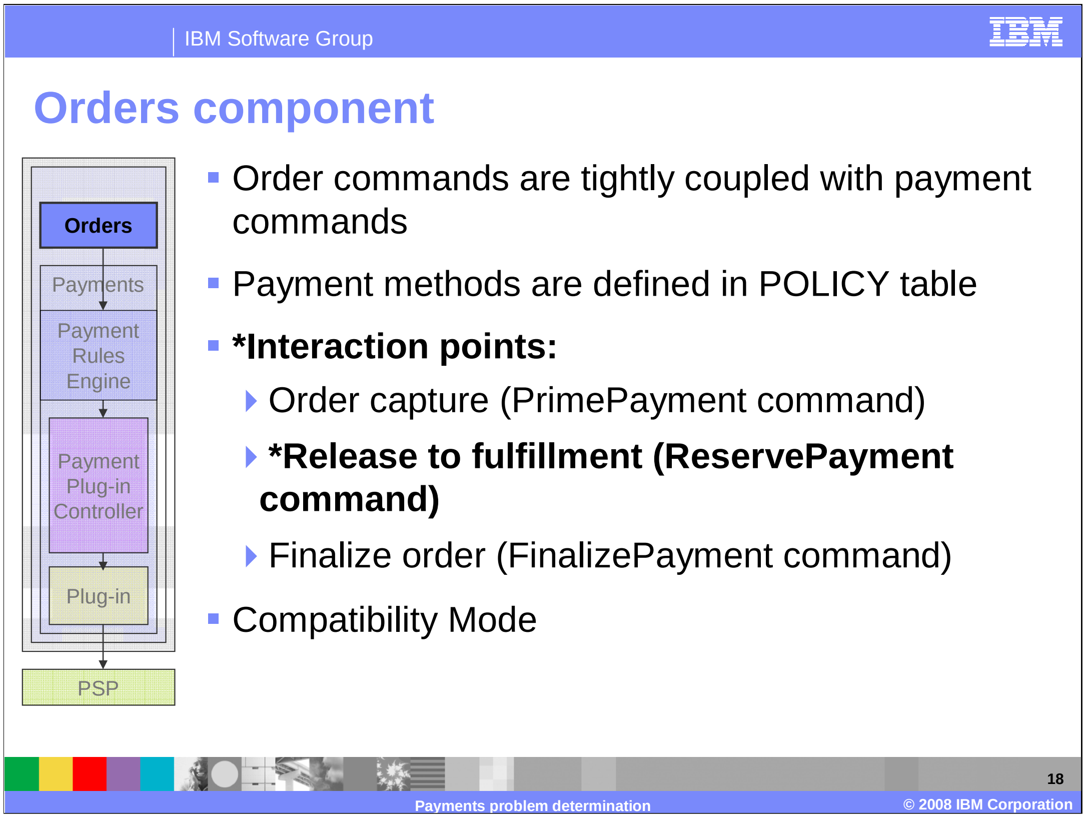
Orders component

Orders
Payments
Payment Rules Engine
Payment Plug-in Controller
Plug-in
PSP

- Order commands are tightly coupled with payment commands
- Payment methods are defined in POLICY table
- ***Interaction points:**
  - ***Order capture (PrimePayment command)**
  - Release to fulfillment (ReservePayment command)
  - Finalize order (FinalizePayment command)
- Compatibility Mode

There are three places where order commands invoke payment commands.

First is during order capture. This is when a shopper initially submits an order to WebSphere Commerce. At this point, the shopper might need to get payment approval. To achieve payment approval, the order command talks to the payments component by using a task command called PrimePayment.



IBM Software Group

Orders component

Orders

Payments

Payment Rules Engine

Payment Plug-in Controller

Plug-in

PSP

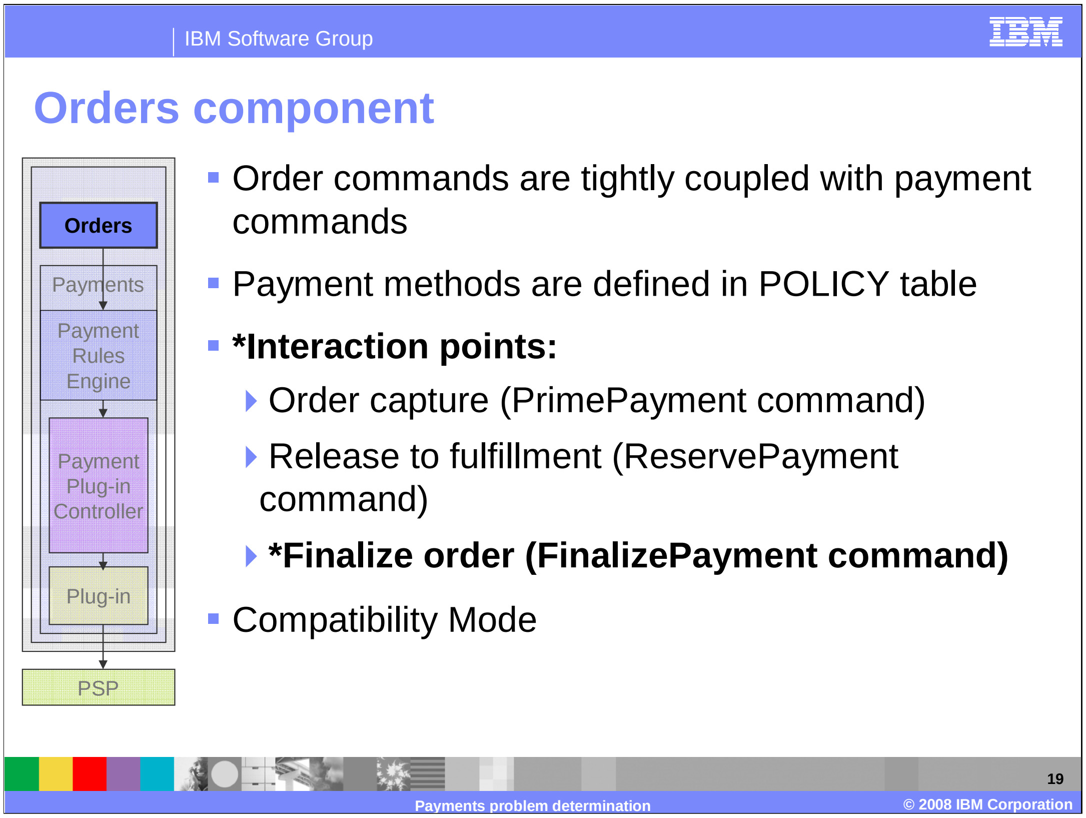
- Order commands are tightly coupled with payment commands
- Payment methods are defined in POLICY table
- ***Interaction points:**
  - Order capture (PrimePayment command)
  - ***Release to fulfillment (ReservePayment command)**
  - Finalize order (FinalizePayment command)
- Compatibility Mode

18

Payments problem determination

© 2008 IBM Corporation

The second case occurs during fulfillment, which is when the order is ready to be shipped. Some businesses choose to deposit the funds at this point, which causes an interaction with Payments using the ReservePayment task command.



IBM Software Group

Orders component

Orders
Payments
Payment Rules Engine
Payment Plug-in Controller
Plug-in
PSP

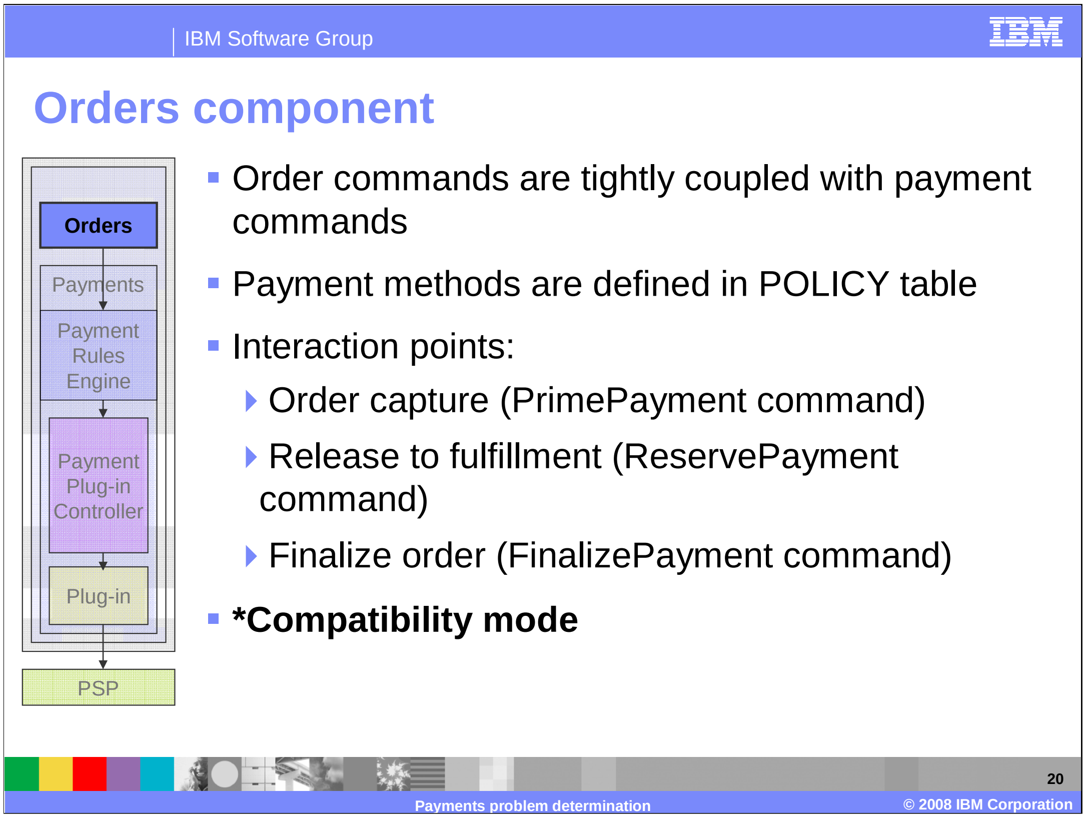
- Order commands are tightly coupled with payment commands
- Payment methods are defined in POLICY table
- ***Interaction points:**
  - Order capture (PrimePayment command)
  - Release to fulfillment (ReservePayment command)
  - ***Finalize order (FinalizePayment command)**
- Compatibility Mode

19

Payments problem determination

© 2008 IBM Corporation

The third scenario occurs when an order is finalized. If payment has not been deposited yet, this is likely where this action takes place. The FinalizePayment is invoked at this point to perform this action.



IBM Software Group

Orders component

Orders
Payments
Payment Rules Engine
Payment Plug-in Controller
Plug-in
PSP

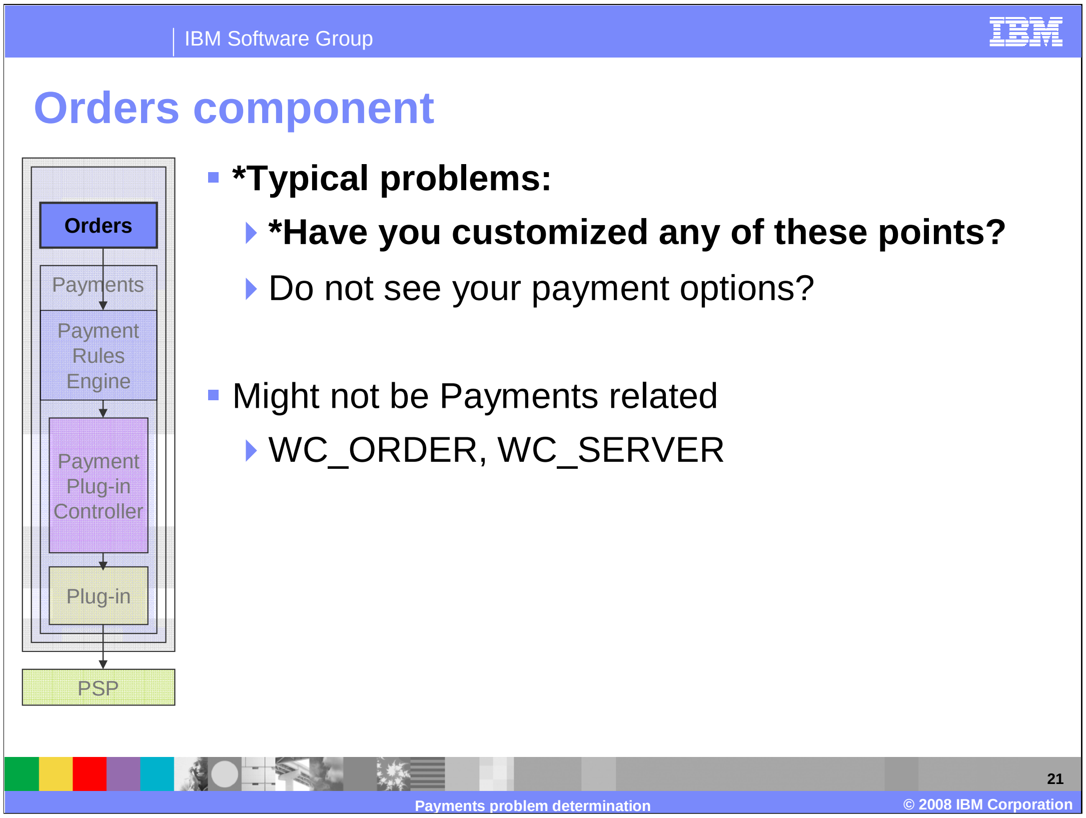
- Order commands are tightly coupled with payment commands
- Payment methods are defined in POLICY table
- Interaction points:
  - Order capture (PrimePayment command)
  - Release to fulfillment (ReservePayment command)
  - Finalize order (FinalizePayment command)
- ***Compatibility mode**

20

Payments problem determination

© 2008 IBM Corporation

Note that the interaction points reference task commands from the version 6.0 payments plug-in framework. However, WebSphere Commerce version 6.0 also provides the ability to use the version 5.6 payment cassette architecture in compatibility mode. Using compatibility mode allows your version 6.0 orders command to directly invoke the payments task commands that interact with a payment manager instance. Note that this presentation does not cover problem determination for the version 5.6 payments cassettes architecture. Refer to the WebSphere Commerce Information Center for details on payment cassettes and the use of compatibility mode.



IBM Software Group

Orders component

Orders

Payments

Payment Rules Engine

Payment Plug-in Controller

Plug-in

PSP

- ***Typical problems:**
  - ***Have you customized any of these points?**
  - Do not see your payment options?
- Might not be Payments related
  - WC_ORDER, WC_SERVER

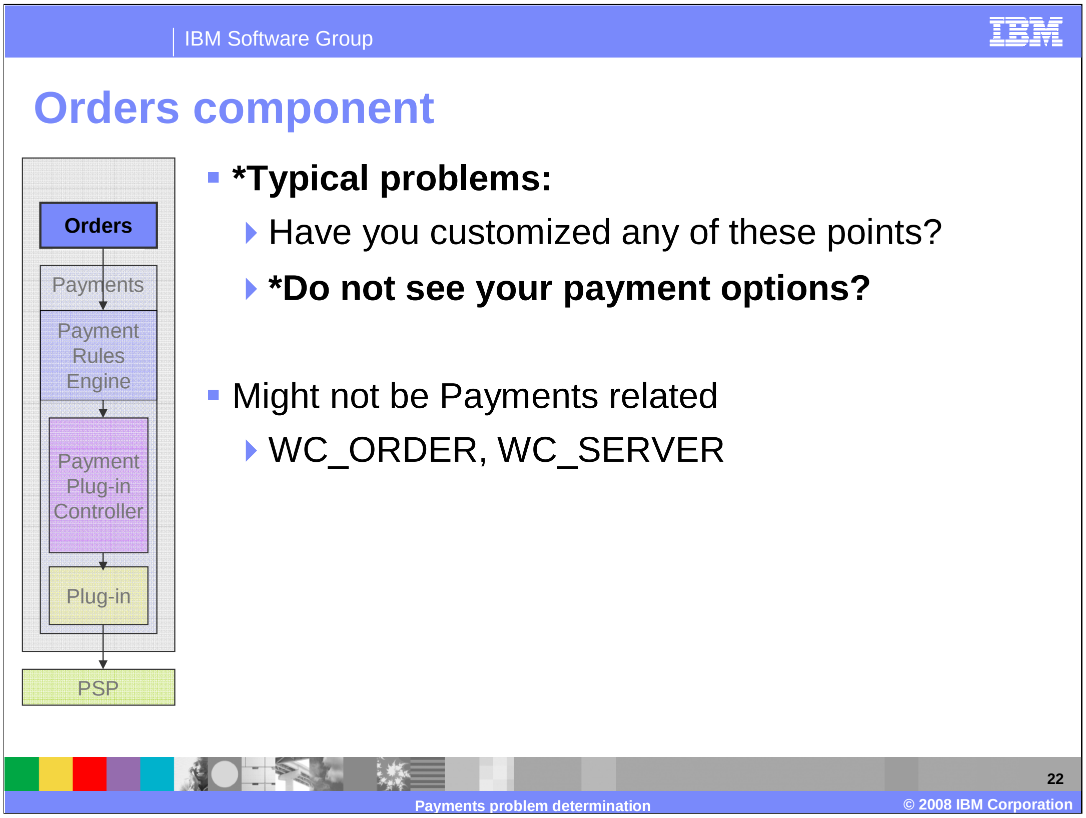
21

Payments problem determination

© 2008 IBM Corporation

Now you understand where the interaction points are, you are ready to understand the typical problems that occur.

Each vendor chooses to handle their payment processing differently. WebSphere Commerce version 6 is designed with the ability to allow users to customize these interaction points so that they can behave appropriately for your business. However, these are where most problems occur. Customizing order commands that invoke these interaction points are the typical cause for differences or changes in expected behavior. Therefore it is critical to understand exactly what has been customized, so that you can identify whether the behavior you are seeing matches the processing flow described in the previous slides.



IBM Software Group

Orders component

Orders

Payments

Payment Rules Engine

Payment Plug-in Controller

Plug-in

PSP

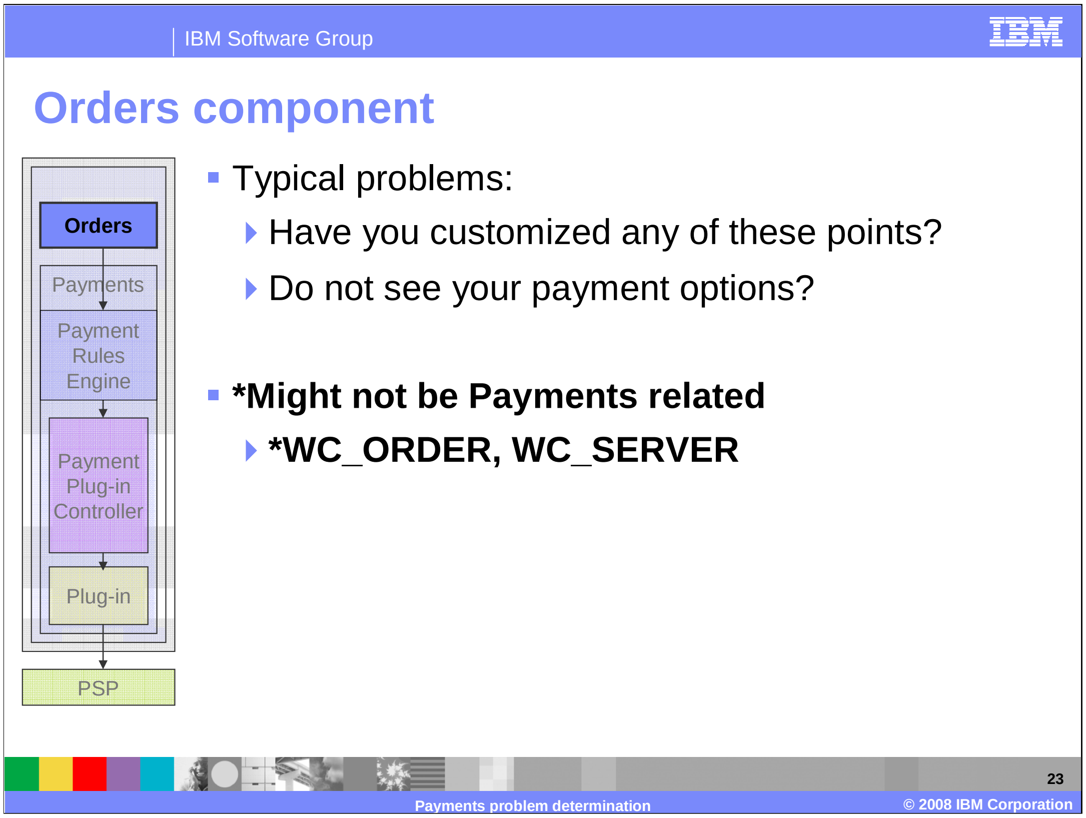
- ***Typical problems:**
  - Have you customized any of these points?
  - ***Do not see your payment options?**
- Might not be Payments related
  - WC_ORDER, WC_SERVER

22

Payments problem determination

© 2008 IBM Corporation

Another common problem is not being able to see your payment options in the store. Specifically, the various payment methods, such as VISA or MASTERCARD, are not being offered to the shopper during checkout. This can happen if the policy information discussed earlier is not configured properly. Ensure that the policy is defined for your store and that the DISPLAY flag in the properties column of the POLICY table is set to true.

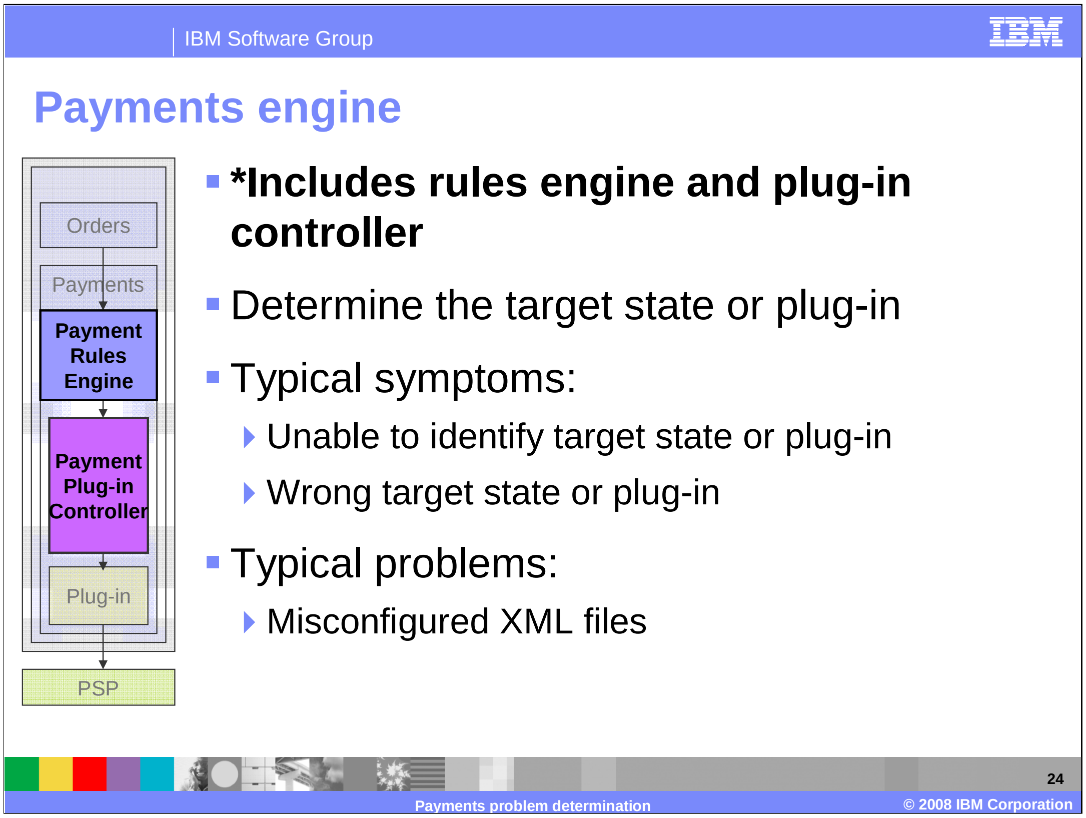


Orders component

Orders → Payments → Payment Rules Engine → Payment Plug-in Controller → Plug-in → PSP

- Typical problems:
  - Have you customized any of these points?
  - Do not see your payment options?
- ***Might not be Payments related**
  - ***WC_ORDER, WC_SERVER**

In many cases, the problem might be happening before Payments is invoked. Therefore, it is important to begin looking at the problem from the orders perspective. Check if Payments is being provided with appropriate information to process the request. In such cases, enabling WC_ORDER and WC_SERVER tracing helps to narrow down the issue.



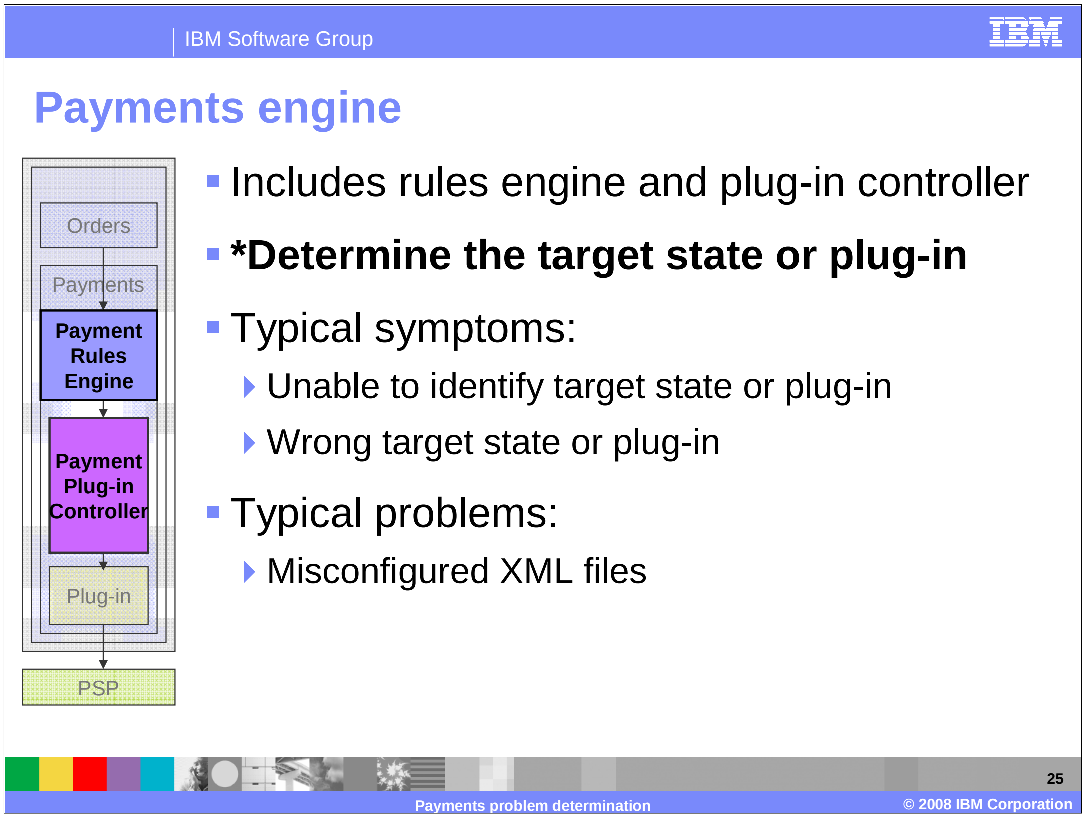
Payments engine

Orders
Payments
Payment Rules Engine
Payment Plug-in Controller
Plug-in
PSP

- ***Includes rules engine and plug-in controller**
- Determine the target state or plug-in
- Typical symptoms:
  - Unable to identify target state or plug-in
  - Wrong target state or plug-in
- Typical problems:
  - Misconfigured XML files

Once you eliminate the Orders component as the cause of the problem, the next component to consider is the payments engine.

The term **payments engine** refers to both the Payment rules engine and the Payment plug-in controller.



Payments engine

Orders

Payments

Payment Rules Engine

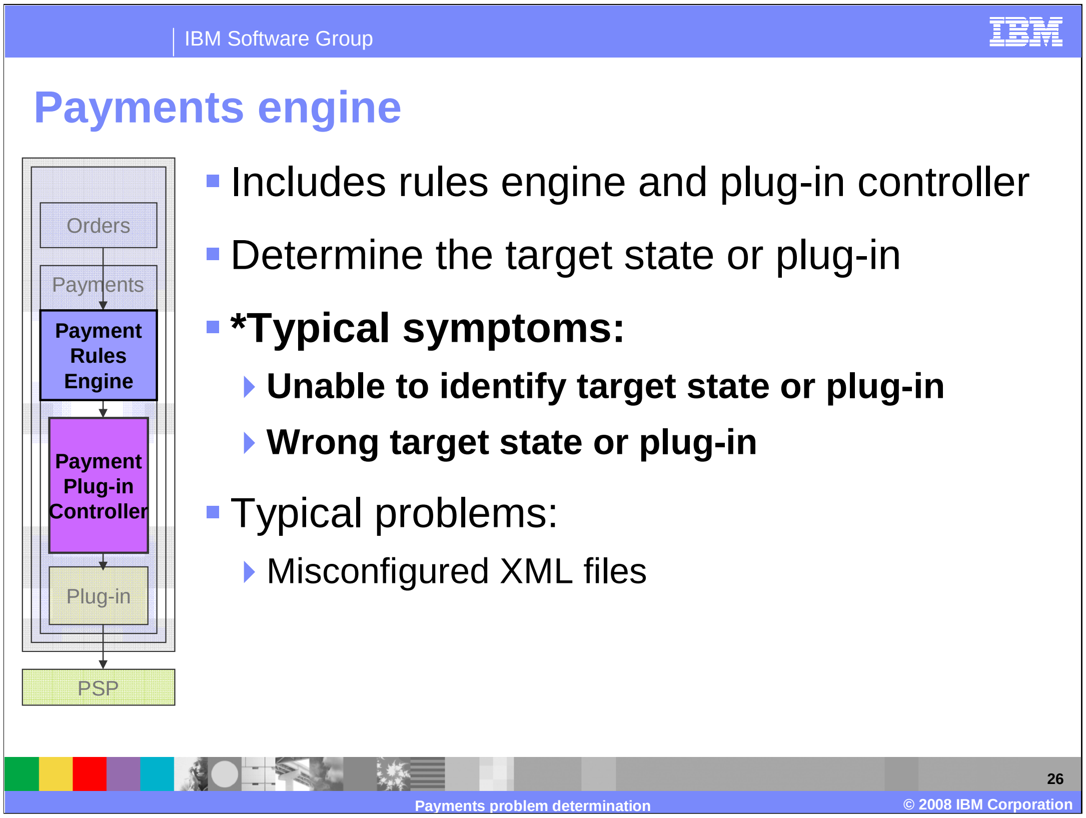
Payment Plug-in Controller

Plug-in

PSP

- Includes rules engine and plug-in controller
- ***Determine the target state or plug-in**
- Typical symptoms:
  - Unable to identify target state or plug-in
  - Wrong target state or plug-in
- Typical problems:
  - Misconfigured XML files

The rules engine and plug-in controller work together to determine the target payment state and plug-in. The payment engine uses the payment method and the interaction points to figure out the target state and plug-in. The interaction points include PrimePayment, ReservePayment, and FinalizePayment.



IBM Software Group

Payments engine

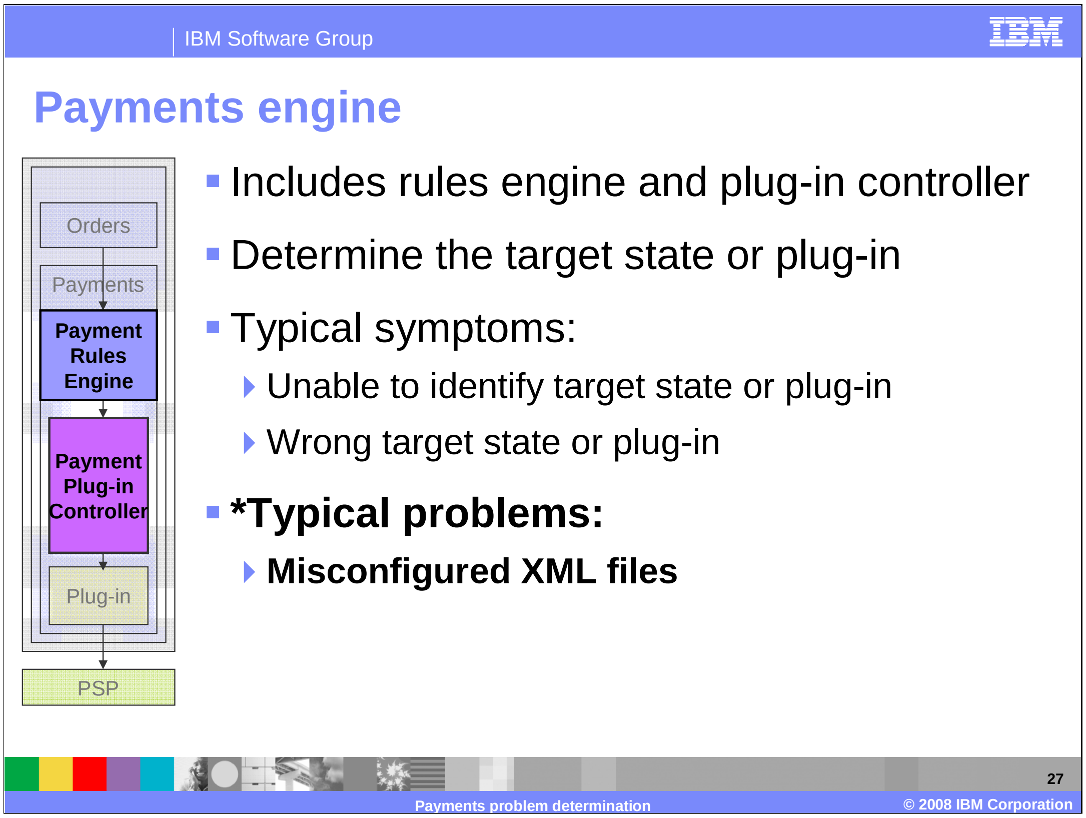
- Includes rules engine and plug-in controller
- Determine the target state or plug-in
- ***Typical symptoms:**
  - **Unable to identify target state or plug-in**
  - **Wrong target state or plug-in**
- Typical problems:
  - Misconfigured XML files

26

Payments problem determination

© 2008 IBM Corporation

Since the payment engine is responsible for determining the target state and plug-in, typical symptoms include failure to identify this information or possibly returning the wrong information. For example, if the PrimePayment command is invoked, the target state is typically **approved**. However, if there is a misconfiguration, the payment engine might conclude that the target state is something else, such as **deposit.** Or the payment engine might create an error indicating that a target state can not be found. The payment engine might not be able to locate a suitable payment plug-in, or return the wrong plug-in.



IBM Software Group

Payments engine

Orders

Payments

Payment Rules Engine

Payment Plug-in Controller

Plug-in

PSP

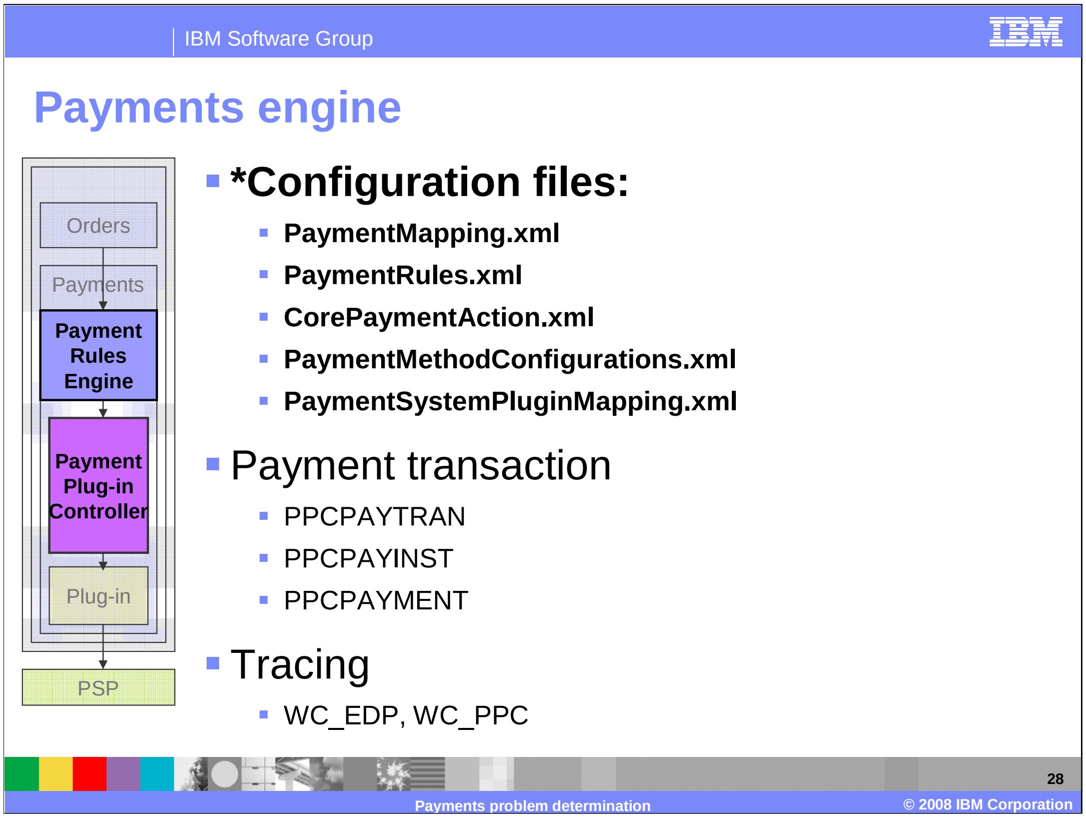
- Includes rules engine and plug-in controller
- Determine the target state or plug-in
- Typical symptoms:
  - Unable to identify target state or plug-in
  - Wrong target state or plug-in
- ***Typical problems:**
  - **Misconfigured XML files**

27

Payments problem determination

© 2008 IBM Corporation

In such cases, the most likely root cause is misconfigured XML files. Because of the need to allow for customization of the payment processing flow, the target state and plug-in are all configurable options inside a series of XML files that govern how the processing flow works. In many cases, changes made within these files are the root cause of many of the above symptoms. It is important to understand what the changes are, why they were made, and what the difference is between the correct behavior versus the problematic behavior.



IBM Software Group

Payments engine

Orders

Payments

Payment Rules Engine

Payment Plug-in Controller

Plug-in

PSP

- ***Configuration files:**
  - **PaymentMapping.xml**
  - **PaymentRules.xml**
  - **CorePaymentAction.xml**
  - **PaymentMethodConfigurations.xml**
  - **PaymentSystemPluginMapping.xml**
- Payment transaction
  - PPCPAYTRAN
  - PPCPAYINST
  - PPCPAYMENT
- Tracing
  - WC_EDP, WC_PPC

28

Payments problem determination

You should examine several things to ensure your configuration is correct.

First, look at the configuration files. These are the five XML files that govern the payment engine.

If you see any of the previously mentioned symptoms, there is a good chance that one of these files has been incorrectly modified.

Each one of these files is discussed in more detail in the payments technical overview presentation. Here is a brief summary of each of these configuration files.

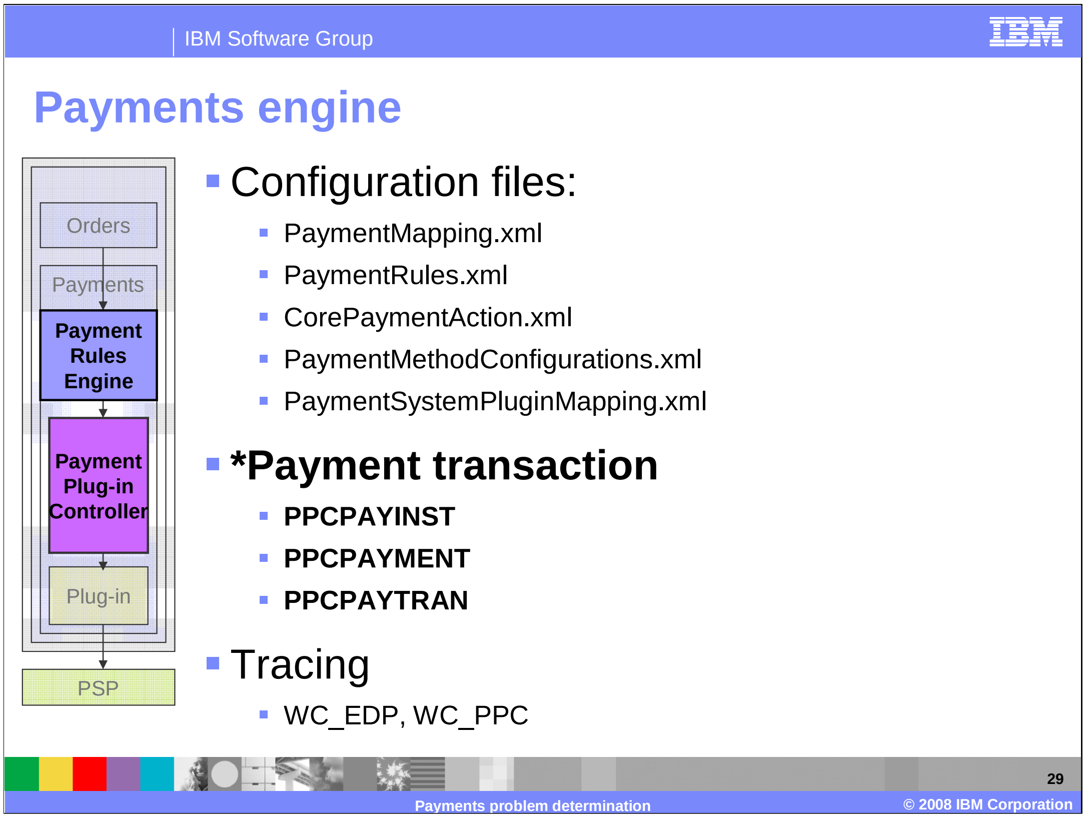
The PaymentMapping.xml is used to determine the payment configuration and payment action rule.

The PaymentRules.xml is used to map the payment rule to the target state.

The CorePaymentAction.xml is used to determine the action required to achieve the target state.

The PaymentMethodConfigurations.xml tells the plug-in controller what the payment system name is based on the payment configuration.

The PaymentSystemPluginMapping.xml maps the payment system name to the appropriate plug-in.



IBM Software Group

Payments engine

Orders
Payments
Payment Rules Engine
Payment Plug-in Controller
Plug-in
PSP

- Configuration files:
  - PaymentMapping.xml
  - PaymentRules.xml
  - CorePaymentAction.xml
  - PaymentMethodConfigurations.xml
  - PaymentSystemPluginMapping.xml
- ***Payment transaction**
  - **PPCPAYINST**
  - **PPCPAYMENT**
  - **PPCPAYTRAN**
- Tracing
  - WC_EDP, WC_PPC

29

Payments problem determination

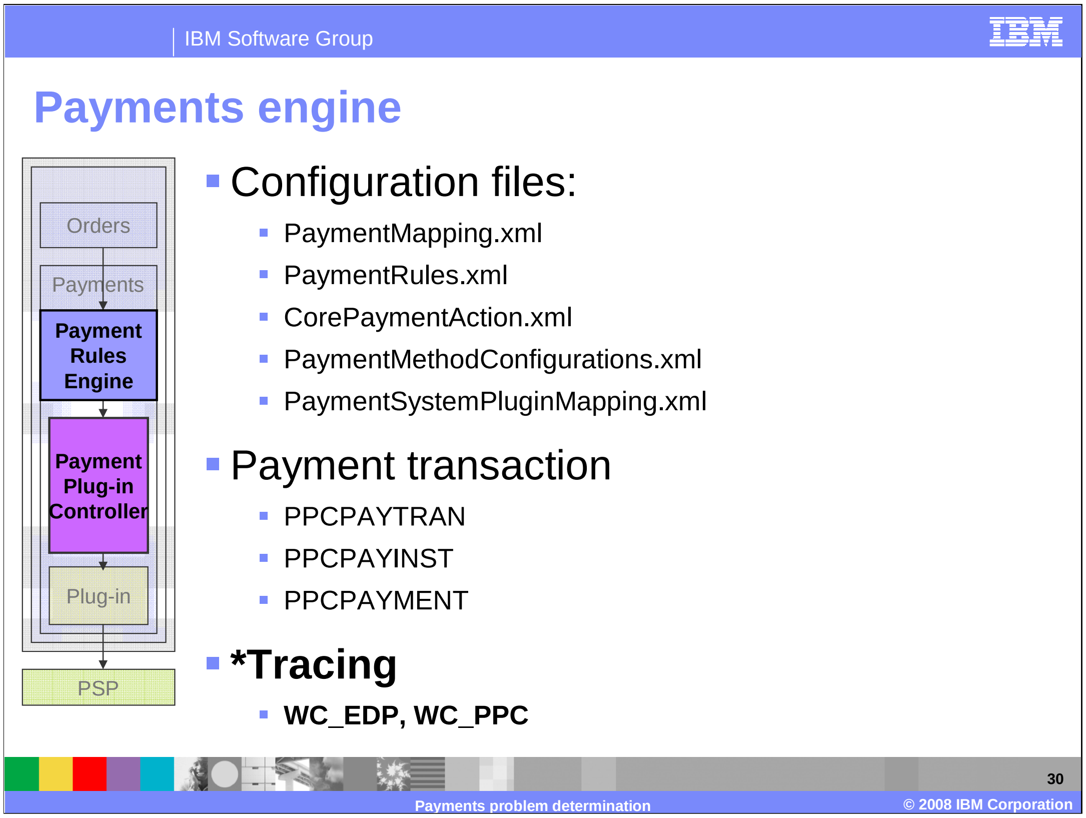
In order to better understand the problem, you should examine the state of the payment transaction. The payment plug-in controller is responsible for creating the payment transaction object that is used to hold the relevant information. This payment transaction is then persisted to the database along with the relevant payment instructions. If something is wrong with the transaction, for example the payment amount, action, or state, then this is the place to check.

Let's take a look at the three tables that store this data:

The PPCPAYINST table stores the payment instruction such as the account number and the amount to approve, deposit, or credit.

The PPCPAYMENT table stores the payment object that is associated with the payment instruction.

The PPCPAYTRAN table stores the financial transaction object. It is likely to have multiple payment transaction objects per payment instruction. For example, APPROVE and DEPOSIT represent two different transactions.



IBM Software Group

Payments engine

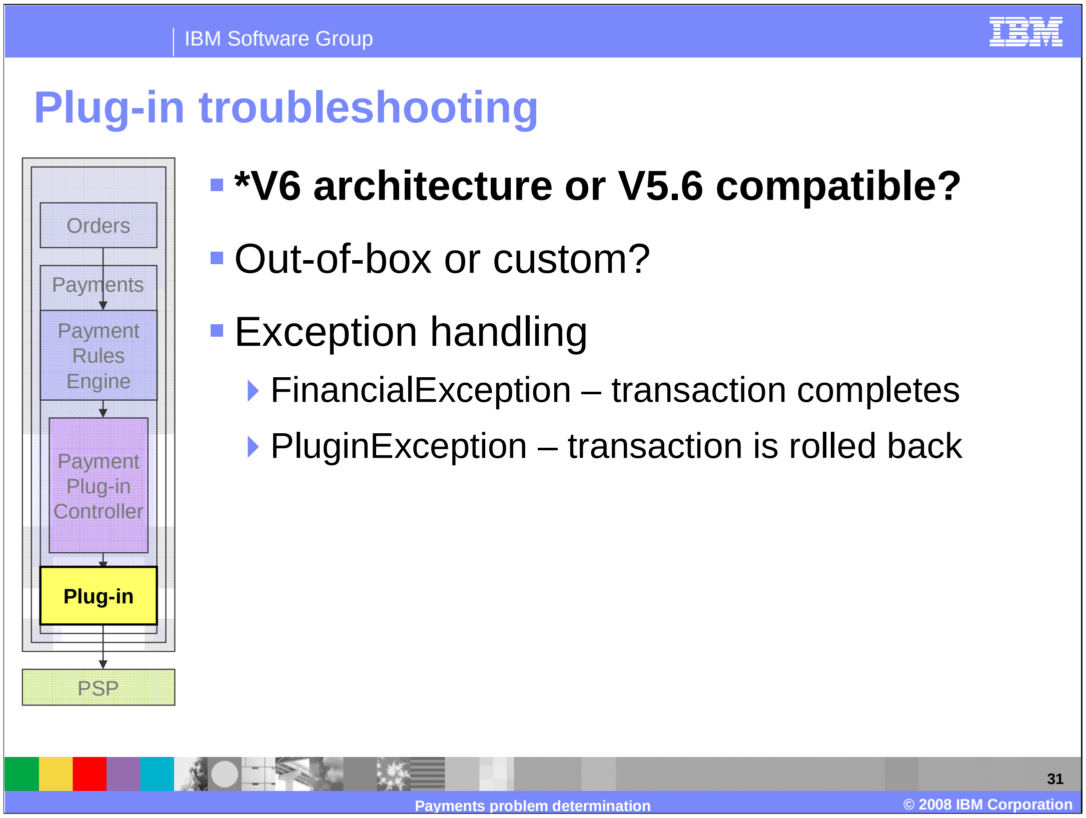
Orders
Payments
Payment Rules Engine
Payment Plug-in Controller
Plug-in
PSP

- Configuration files:
  - PaymentMapping.xml
  - PaymentRules.xml
  - CorePaymentAction.xml
  - PaymentMethodConfigurations.xml
  - PaymentSystemPluginMapping.xml
- Payment transaction
  - PPCPAYTRAN
  - PPCPAYINST
  - PPCPAYMENT
- ***Tracing**
  - **WC_EDP, WC_PPC**

30

Payments problem determination

Finally, if you need additional information about this component, turn on the WC_EDP and WC_PPC tracing.



IBM Software Group

Plug-in troubleshooting

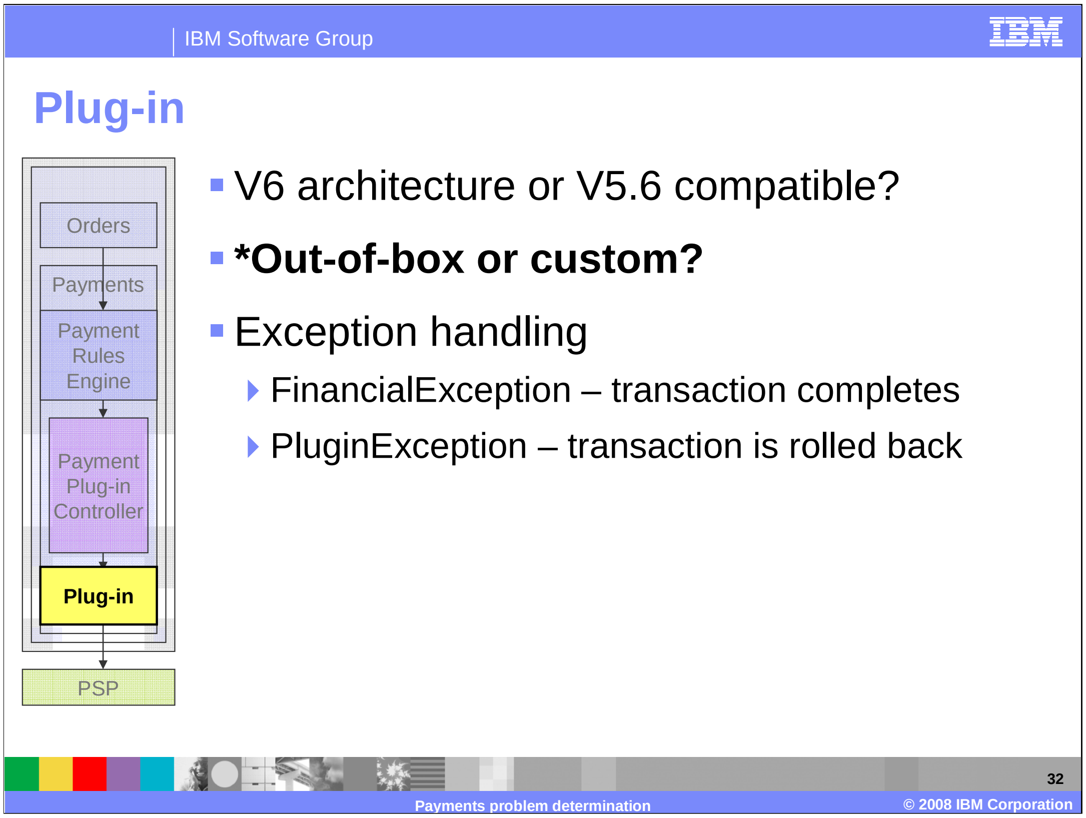
- ***V6 architecture or V5.6 compatible?**
- Out-of-box or custom?
- Exception handling
  - FinancialException – transaction completes
  - PluginException – transaction is rolled back

31

Payments problem determination

This section shows you how to troubleshoot a problem with the Payments plug-in.

First determine your payments compatibility type. If you are using a plug-in for the WebSphere Commerce Payments version 6 architecture, you can proceed with the troubleshooting techniques in subsequent slides. If you are using the provided plug-in designed to communicate with a migrated WebSphere Payment Manager server that provides version 5.6 payment cassette capability, this type is version 5.6 compatible. For the version 5.6 compatible payment type, you should use the version 5.6 approach to troubleshooting payment cassettes. This presentation does not cover problem determination for the previous version 5.6 payments cassettes architecture. Refer to the WebSphere Commerce Information Center for details on how to troubleshoot payment cassettes.



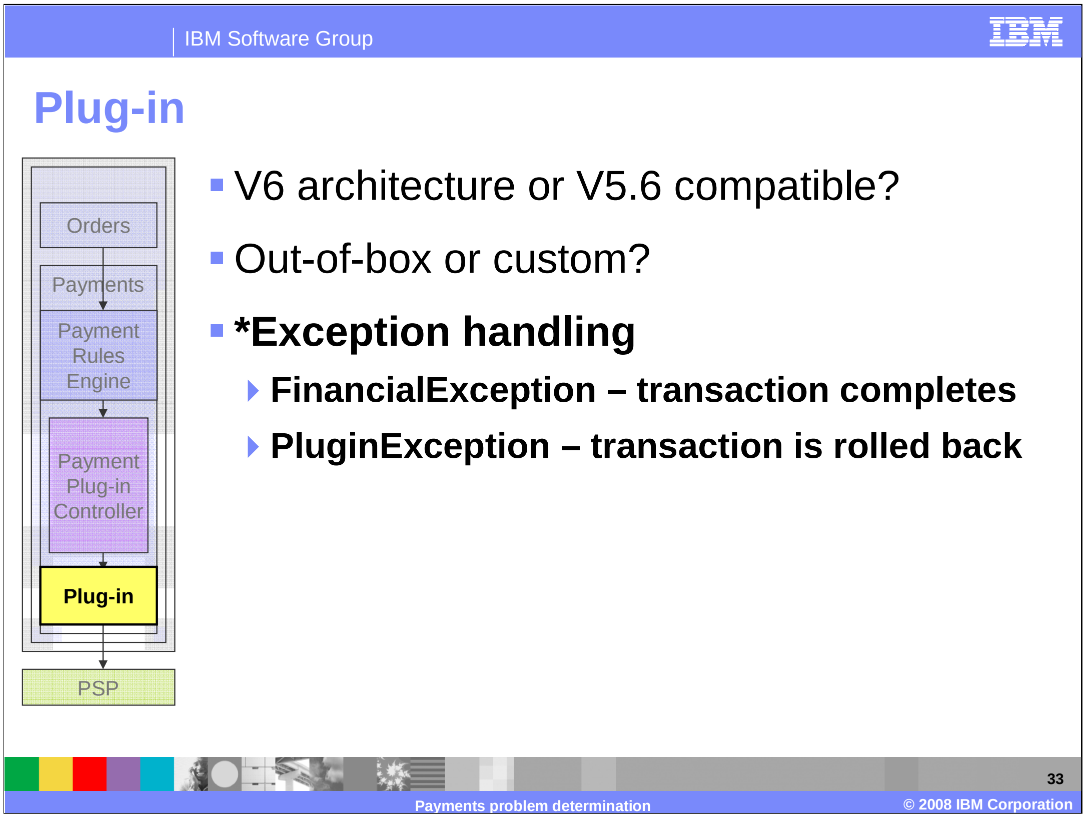
Plug-in

- V6 architecture or V5.6 compatible?
- ***Out-of-box or custom?**
- Exception handling
  - FinancialException – transaction completes
  - PluginException – transaction is rolled back

If you have determined that you are using a version 6 architecture plug-in, you should next determine whether this is a standard plug-in or if it has been customized.

If the plug-in uses one of the standard plug-ins provided by WebSphere Commerce, then the situation becomes a little easier to debug because enabling the appropriate plug-in trace provides sufficient information.

However, if the plug-in is one that has been customized, then a good understanding of the exception handling process within the plug-in is critical to troubleshooting a problem.


Orders
Payments
Payment Rules Engine
Payment Plug-in Controller
Plug-in
PSP

IBM Software Group

Plug-in

- V6 architecture or V5.6 compatible?
- Out-of-box or custom?
- ***Exception handling**
  - **FinancialException – transaction completes**
  - **PluginException – transaction is rolled back**

33

Payments problem determination

© 2008 IBM Corporation

Exception handling can be broken up into two types.

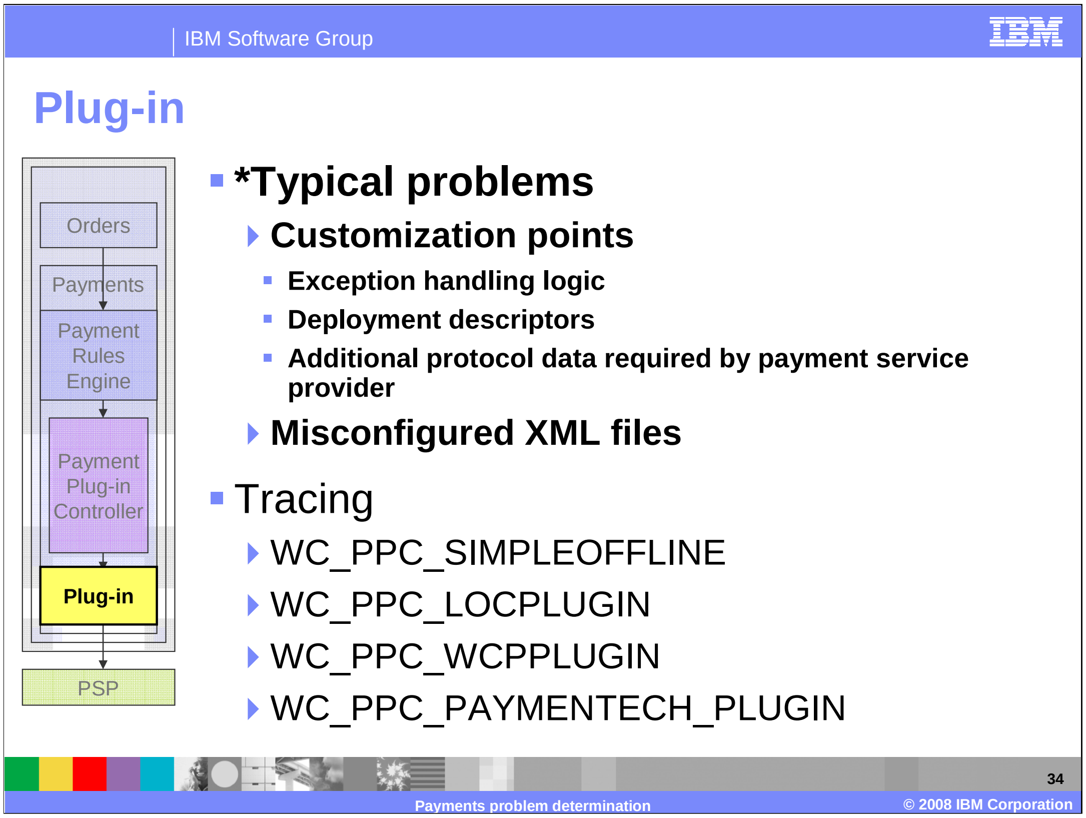
The first type is a FinancialException. This exception does not prevent the transaction from completing. That is, the order can still be submitted successfully.

Your custom plug-in might create a FinancialException in certain cases. For example, if the payment service provider fails to respond to your plug-in in a timely manner, you can still accept the Order and confirm it later manually.

The second type is a PluginException, which causes the transaction to fail and roll back. Your plug-in might cause a PluginException if an error occurred somewhere in the business logic.

Understanding the type of exception helps you associate the error with the state of the transaction.

For standard plug-ins provided by WebSphere Commerce, the Information Center provides details about the type of exception that are created based on the payment return code from the backend payment service provider.



Plug-in

Orders
Payments
Payment Rules Engine
Payment Plug-in Controller
Plug-in
PSP

- ***Typical problems**
  - **Customization points**
    - **Exception handling logic**
    - **Deployment descriptors**
    - **Additional protocol data required by payment service provider**
  - **Misconfigured XML files**
- Tracing
  - WC_PPC_SIMPLEOFFLINE
  - WC_PPC_LOCPLUGIN
  - WC_PPC_WCPPLUGIN
  - WC_PPC_PAYMENTECH_PLUGIN

Typical problems encountered in the plug-in are related to customization points and misconfigured XML files.

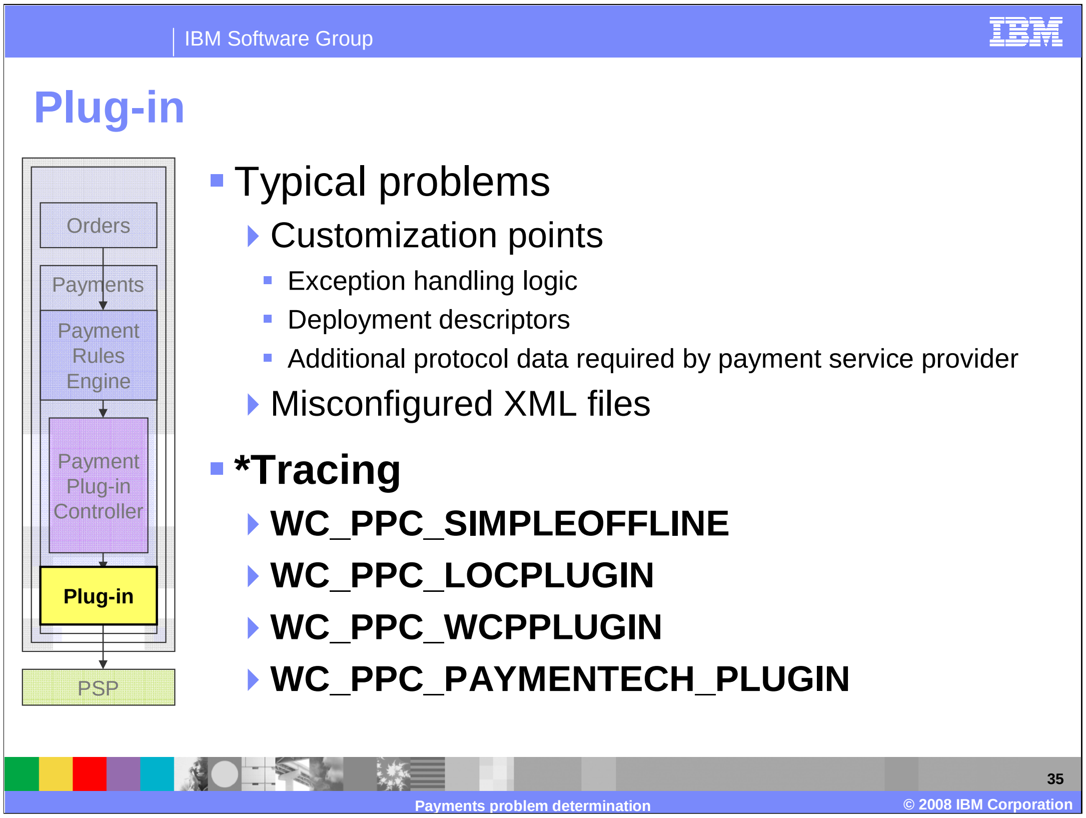
Regarding customization points, pay special attention to these recommendations.

First, review how the exception handling logic is implemented. A wrong type of exception results in completely incorrect behavior of the payment transaction. Ensure the appropriate exception is created.

Second, check your deployment descriptors. Each payment plug-in contains a deployment descriptor that defines the main class so the payment engine knows what to call.

Third, pay attention to any additional protocol data that might be required by your payment service provider. Some payment service providers require additional data to be provided in addition to credit card numbers and amount. Ensure your plug-in is passing all the necessary data to the backend.

Another typical problem is misconfigured XML files. These are the same set of XML files previously described in the payment engine component. Ensure these are properly defined and point to the correct plug-in.



IBM Software Group

Plug-in

- Typical problems
  - Customization points
    - Exception handling logic
    - Deployment descriptors
    - Additional protocol data required by payment service provider
  - Misconfigured XML files
- ***Tracing**
  - **WC_PPC_SIMPLEOFFLINE**
  - **WC_PPC_LOCPLUGIN**
  - **WC_PPC_WCPPLUGIN**
  - **WC_PPC_PAYMENTECH_PLUGIN**

35

Payments problem determination

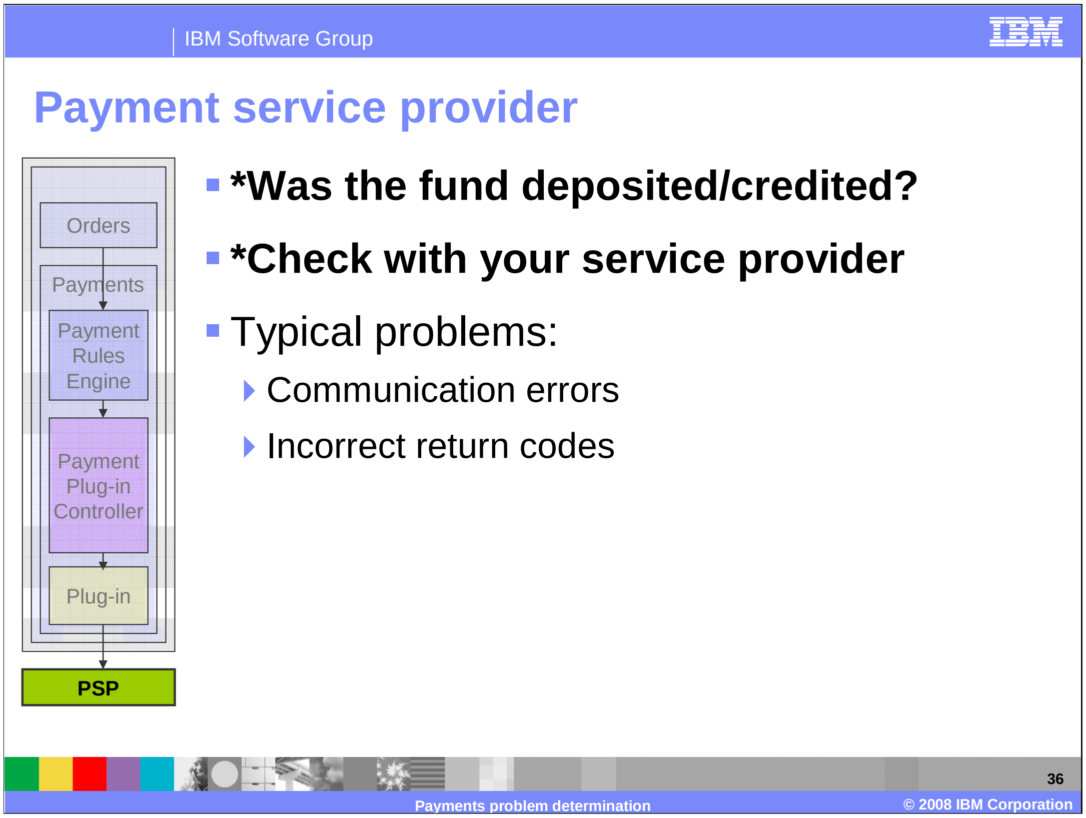
Finally, if further troubleshooting is required, enable the necessary tracing for your specific plug-in. The trace setting depends on the type of plug-in you are using. Here is a list of available traces :

The WC_PPC_SIMPLEOFFLINE trace is used for the simple offline plug-in.

The WC_PPC_LOCPLUGIN trace is used for the line of credit plug-in.

The WC_PPC_WCPPLUGIN trace is used for the WebSphere Commerce Payments plug-in which communicates with a V5.6 payment manager.

The WC_PPC_PAYMENTECH_PLUGIN trace is used for the Paymentech plug-in that was added in WebSphere Commerce Feature Pack 2.



Payment service provider

Orders

Payments

Payment Rules Engine

Payment Plug-in Controller

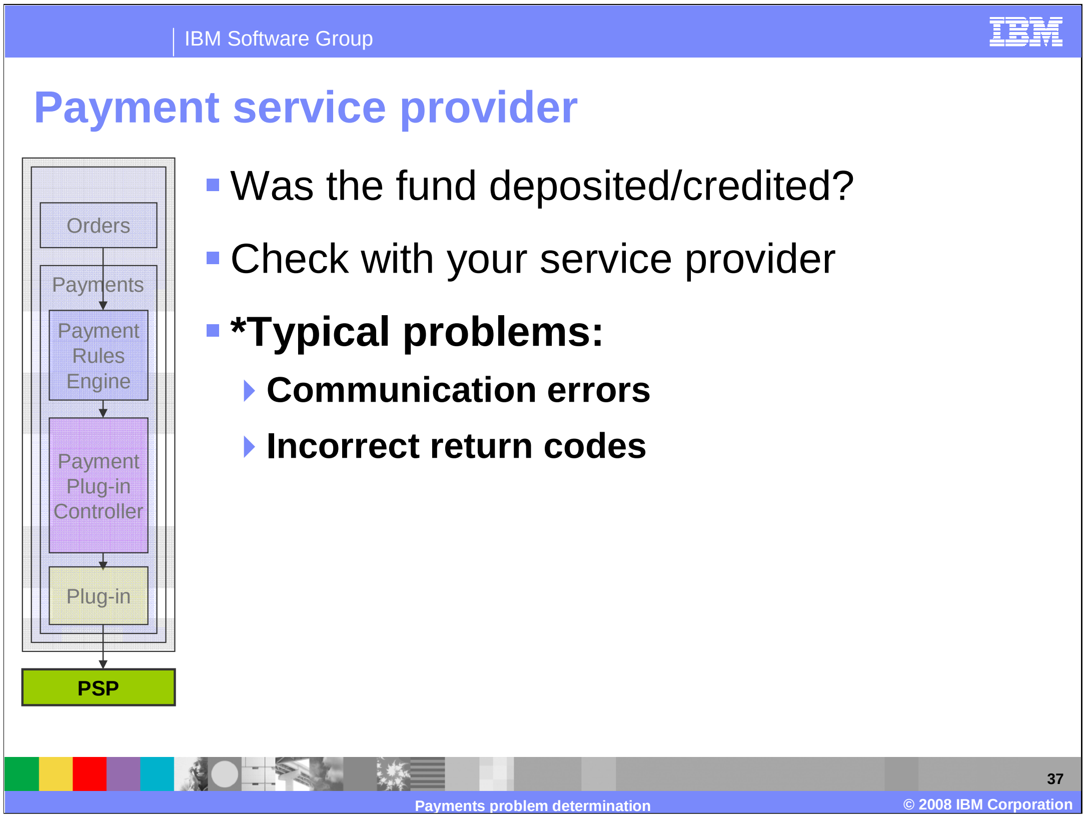
Plug-in

PSP

- ***Was the fund deposited/credited?**
- ***Check with your service provider**
- Typical problems:
  - Communication errors
  - Incorrect return codes

If you have examined all of the above components and still have a problem, then it is very likely the problem is coming from the backend payment service provider.

In such cases, the most important thing to check is whether the funds are properly deposited into your account or credited back to the shopper’s account. This can not be done within WebSphere Commerce. Therefore it is important to contact your payment service provider to ensure the expected actions have been completed on the backend.


Orders
Payments
Payment Rules Engine
Payment Plug-in Controller
Plug-in
PSP

IBM Software Group

Payment service provider

- Was the fund deposited/credited?
- Check with your service provider
- ***Typical problems:**
  - **Communication errors**
  - **Incorrect return codes**

37

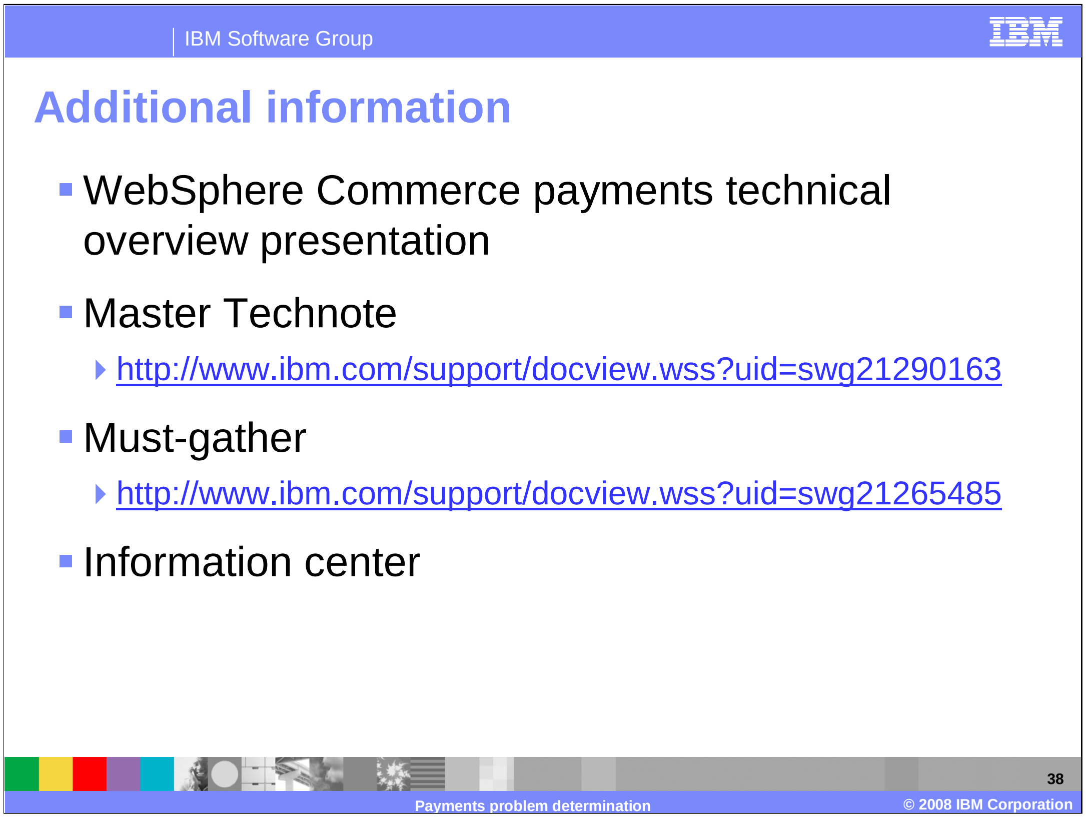
Payments problem determination

© 2008 IBM Corporation

When problems arise with the payment service provider, there are some very common causes for failure of the financial transaction.

The most typical cause is a communication problem between the WebSphere Commerce server and the backend. This might include failed requests, or timeouts waiting for a response. In each of these cases, it is important to contact your payment service provider to ensure the requests have been received. Because the most important aspect here is making sure the money is properly handled.

Another typical scenario is when the payment service provider sends an incorrect return code. These might show up as errors in WebSphere Commerce based on how the plug-in is handling the exception. Unless the return code is obvious, it is always recommended to check with the payment service provider for the reasons behind it.



Additional information

- WebSphere Commerce payments technical overview presentation
- Master Technote
  - http://www.ibm.com/support/docview.wss?uid=swg21290163
- Must-gather
  - http://www.ibm.com/support/docview.wss?uid=swg21265485
- Information center

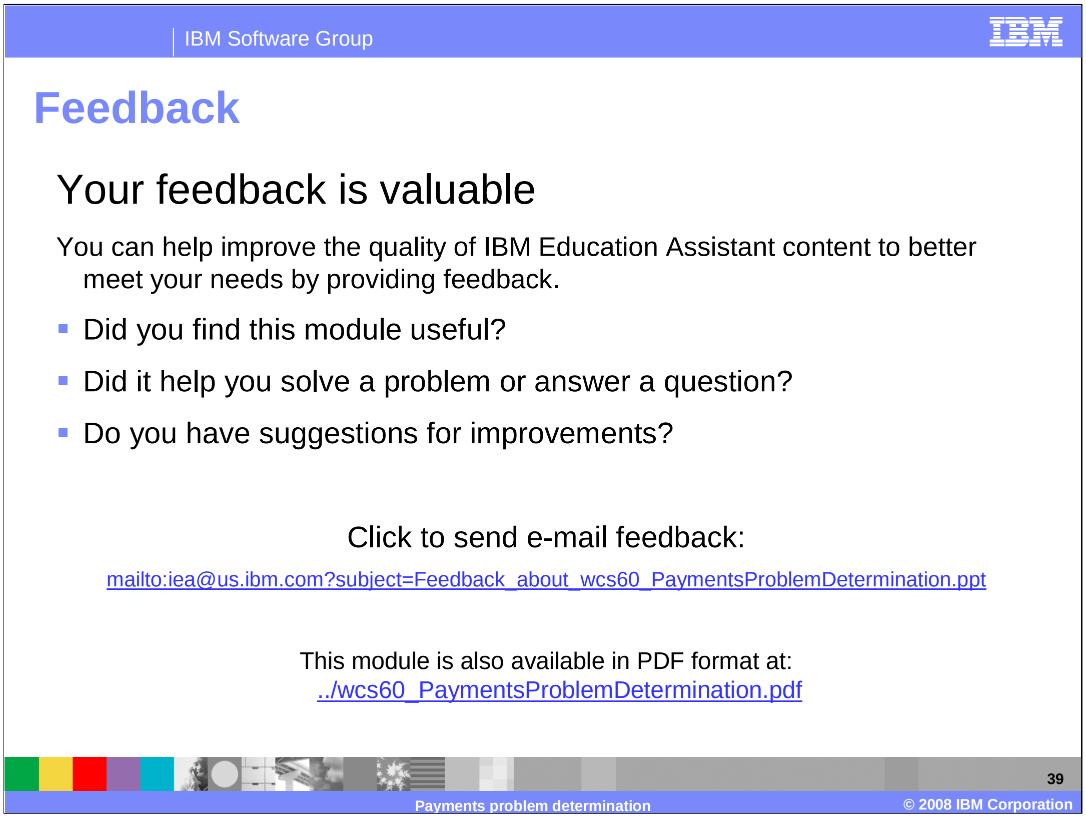
Here is some additional information that can help you understand your WebSphere Commerce payments and troubleshooting.

WebSphere Commerce payments technical overview presentation is available on IBM Education Assistant. This presentation gives you an in-depth look at the entire payment processing flow and how each of the XML files are used governing the payments engine.

The Master Technote lists all of the current known issues with WebSphere Commerce payments. This is a good place to check if your problem already has a solution.

If you are unable to identify the root cause and require assistance from IBM WebSphere Commerce support, having the information ready, as described in the must-gather document, will speed up the problem determination process.

Finally, the WebSphere Commerce Information Center contains tutorials on how to properly setup a payments configuration and provides information about each of the components.



Feedback

Your feedback is valuable

You can help improve the quality of IBM Education Assistant content to better meet your needs by providing feedback.

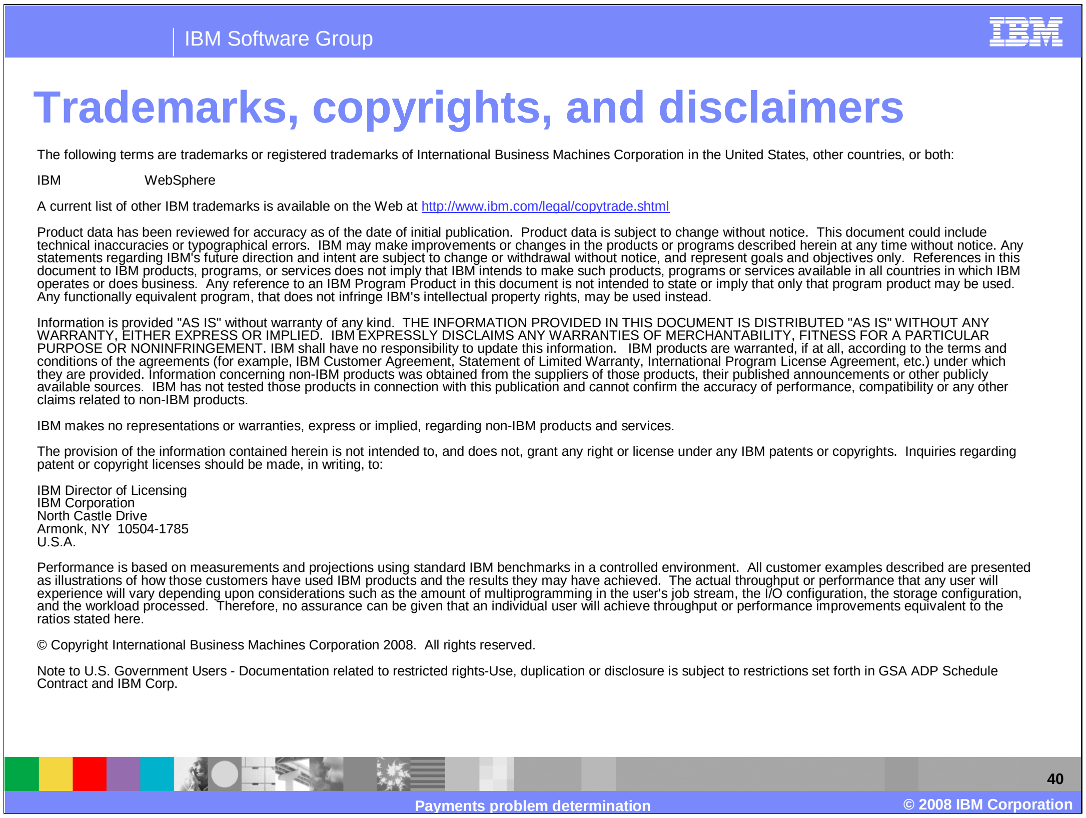
- Did you find this module useful?
- Did it help you solve a problem or answer a question?
- Do you have suggestions for improvements?

Click to send e-mail feedback:

mailto:iea@us.ibm.com?subject=Feedback_about_wcs60_PaymentsProblemDetermination.ppt

This module is also available in PDF format at:
../wcs60_PaymentsProblemDetermination.pdf

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



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	WebSphere

A current list of other IBM trademarks is available on the Web at http://www.ibm.com/legal/copytrade.shtml

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 or changes in the products or programs 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 (for example, 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.