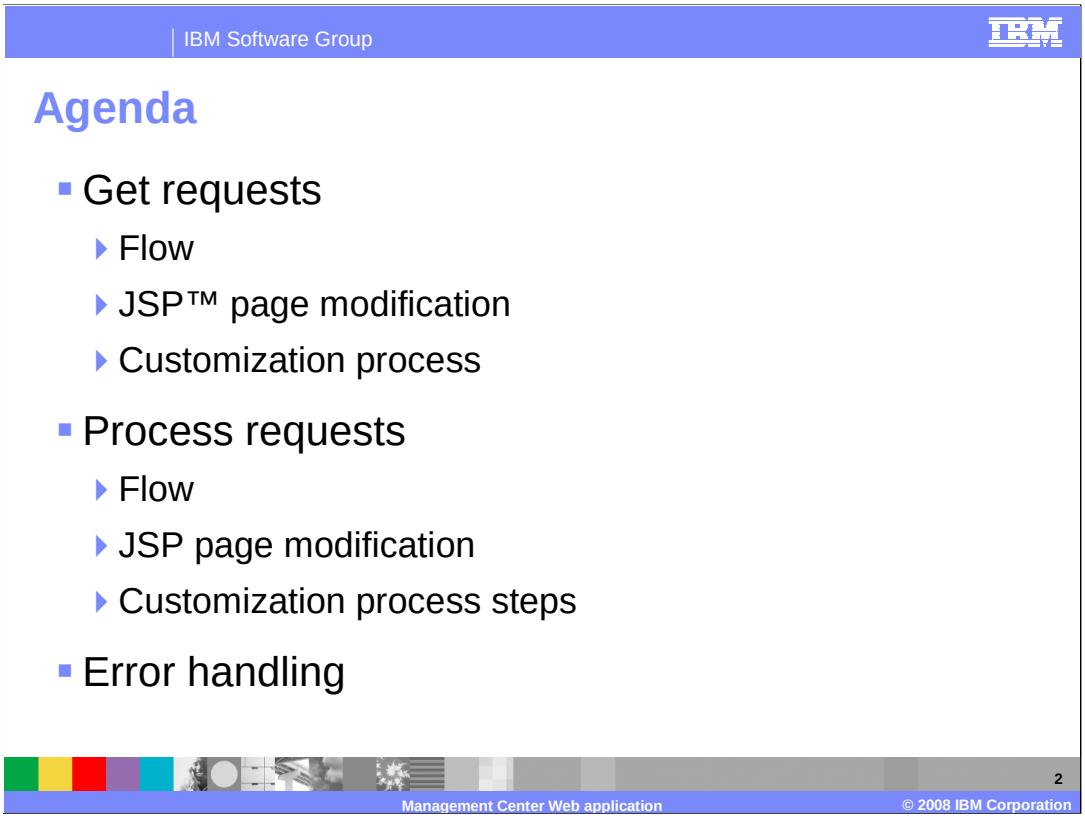
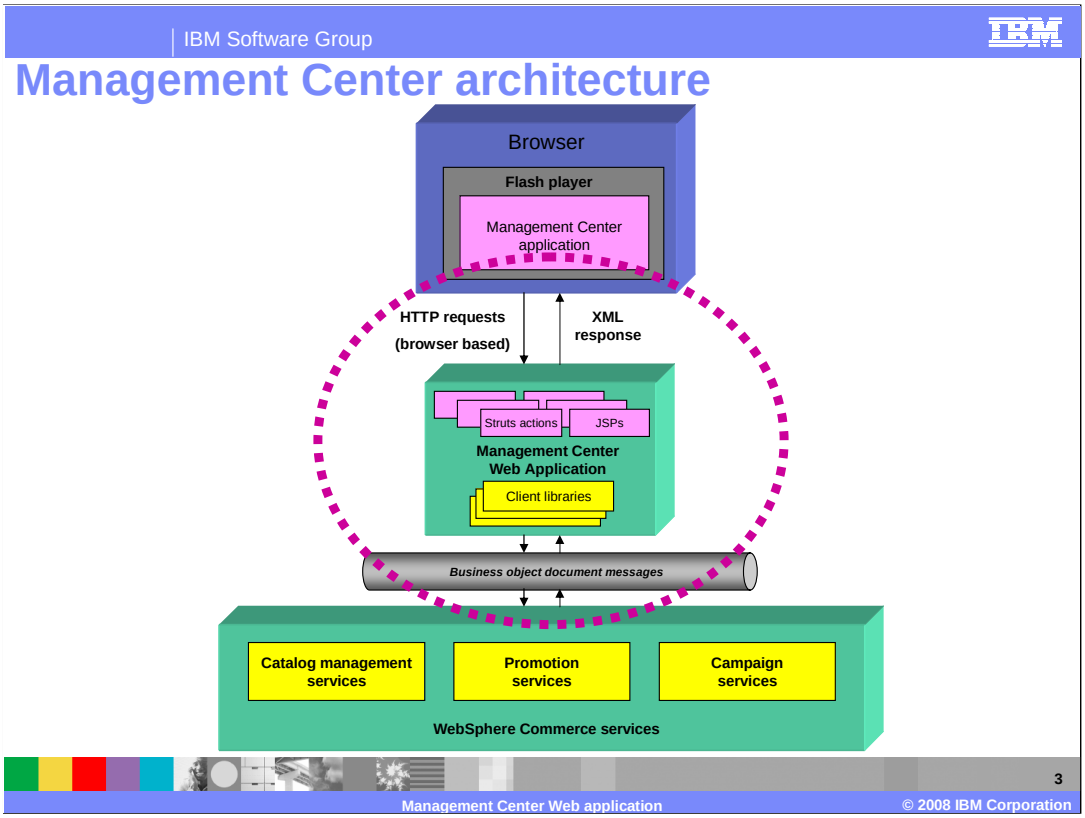
Welcome to the WebSphere Commerce V6 Feature Pack 3 presentation. This presentation describes the Management Center Web application in WebSphere Commerce in Feature Pack 3.



This presentation covers customization of both get requests and process requests and how they are different.

Each request type covers details of request flow, modification of JSP files, and the general customization process steps.

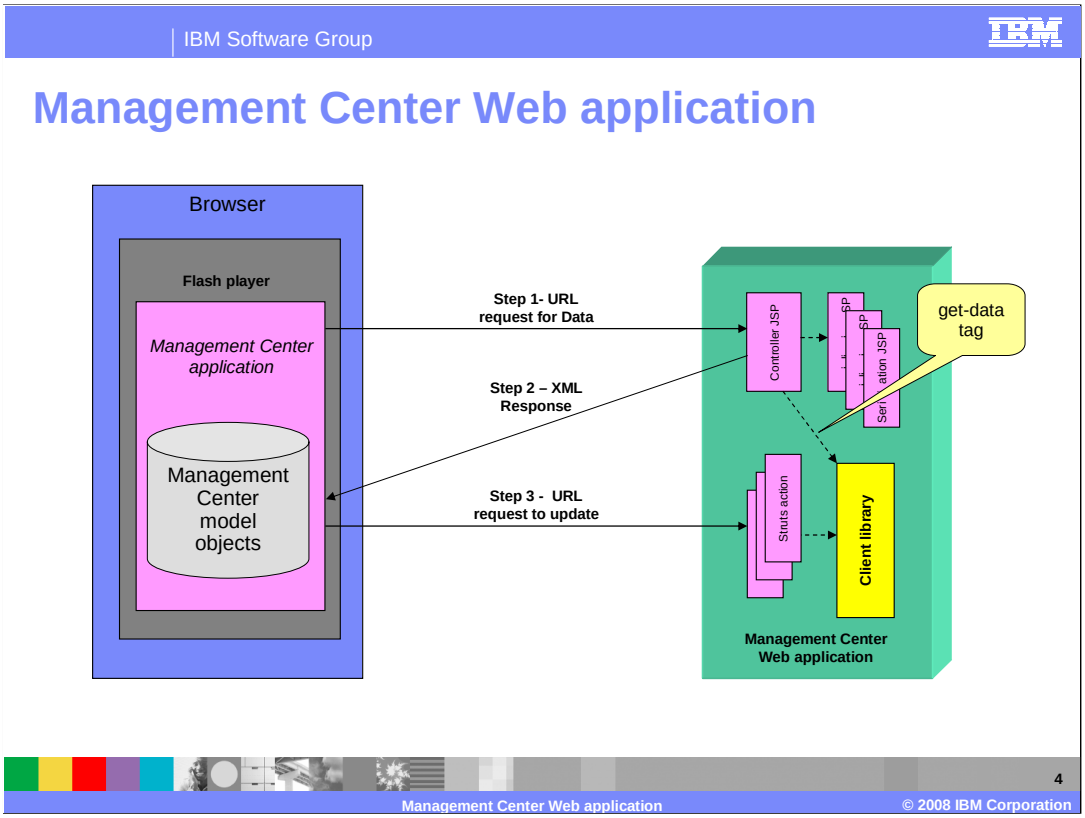
Finally, error handling for any type of request is discussed.



This diagram illustrates a high-level overview of the Management Center framework, along with the types of requests and responses that are passed between the Management Center and WebSphere Commerce. The Management Center Web application is a Struts Web application that facilitates communication between the Management Center and WebSphere Commerce services.

The Management Center sends URL requests and receives XML responses, while the WebSphere Commerce server uses Business Object Document (BOD) messages to retrieve nouns and invoke business logic. The Management Center Web application acts as a mediator between the Management Center and WebSphere Commerce services by converting requests and responses into the appropriate type of data. More specifically, the Management Center Web application provides Struts actions to send data from the Management Center to WebSphere Commerce by converting URL requests into BOD messages. It also returns data to the Management Center from WebSphere Commerce by converting BOD messages into an XML response that the Management Center understands.

The data transferred between the Management Center and the WebSphere Commerce server is called Management Center objects. A Management Center object is an entity that can be created, changed, deleted, or retrieved using the Management Center. Examples of Management Center objects are a category, a product, a kit and a promotion.



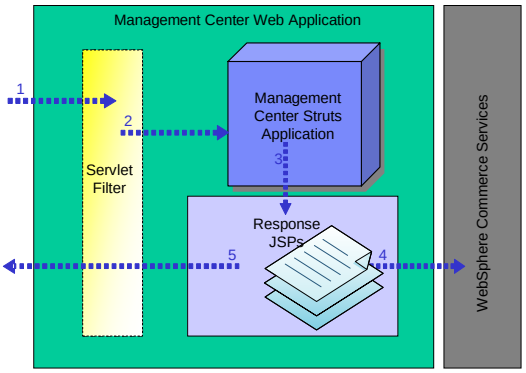
The flow of a URL request from the Management Center to the WebSphere Commerce server differs slightly when retrieving Management Center objects (get) and when processing Management Center objects (create or write).

To retrieve Management Center objects, the Management Center Web application uses two types of JSP™ pages: Get controller JSP pages, and Serialization JSP fragments. These JSP pages facilitate the mediation between the Management Center Web application and WebSphere Commerce services. These JSP pages are not intended to display data to you, for example, through the store front.

To create or write Management Center objects, the Management Center Web application transforms the URL request into a Process BOD or a Change BOD message.

Flow of a Get request

1. The request enters the servlet filter to resolve the authentication information for the request.
2. The Struts action forward is executed.
3. The Struts action forward immediately forwards to the JSP configured in the Struts configuration.
4. The JSP uses the get-data tag to construct and invoke the appropriate Get service.
5. The JSP formats the list of nouns into the XML format that the Management Center recognizes.



This diagram shows the life cycle of a URL request to retrieve Management Center objects.

Step 1: The Management Center issues a URL request. The URL request enters the identity token servlet filter to resolve the authentication information for the request.

Step 2: The Struts servlet resolves the request into an action forward.

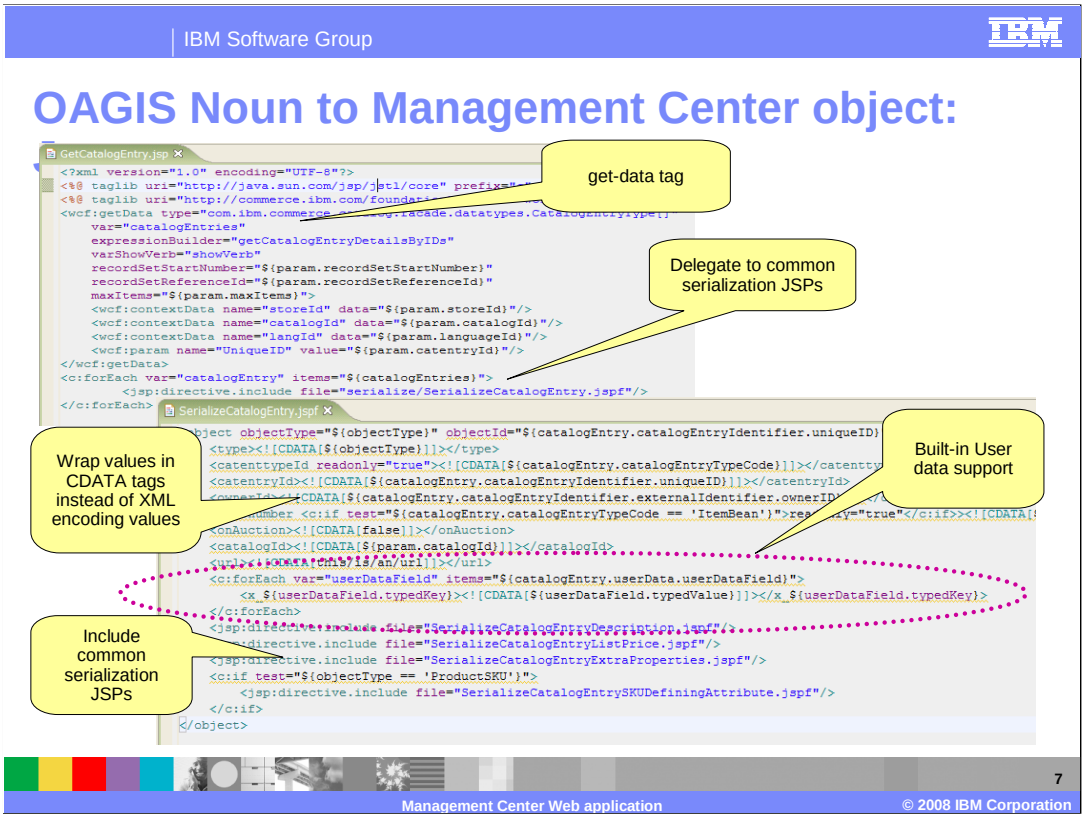
Step 3: The Struts action forward immediately forwards to the Get controller JSP page indicated in the Struts configuration.

Step 4: The Get controller JSP page uses the get-data tag to construct and invoke the appropriate get service to retrieve the list of nouns from the WebSphere Commerce Server that match the expression. The Get controller JSP page then sends the nouns from the response BOD to the serialization JSP fragments.

Step 5: One or more serialization JSP fragments format the nouns into the XML representation that is expected by the Management Center, and then send the XML representation to the Management Center. This is the response of the URL request.

Here is an example of the Struts configuration for Get requests. The Struts actions that handle Get requests are ActionForwards to the corresponding controller JSP page that understands what get-data expression to call to fetch the data and delegate to serialization JSP pages.

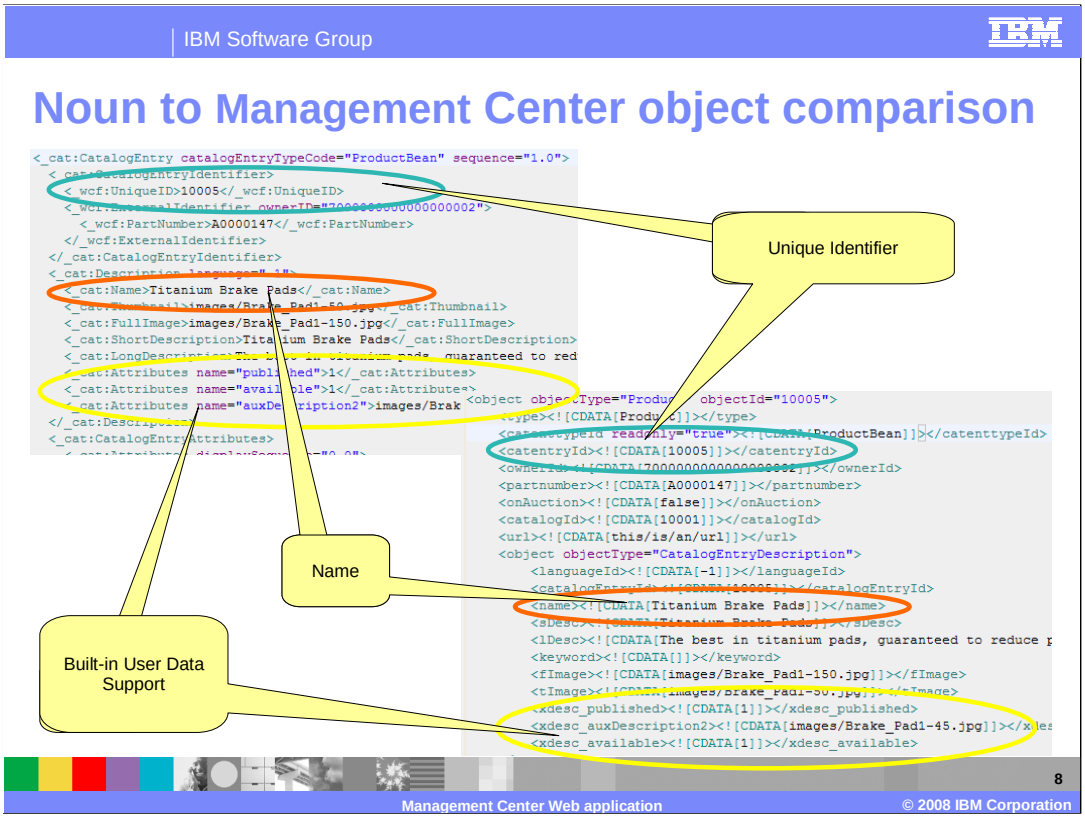
Notice the naming convention used in this example. Get requests should always start with the word Get. In this example it is GetProductChildren where Product can be replaced with any object name. If there are multiple get-children services for a particular object, the scope of the service is also included in the name, prefixed by a dash. In this example, GetProductChildren dash LanguageDescriptions. This convention helps with readability and means, in most cases; you can determine which JSP page you might need to customize by referring to the Struts configuration rather than tracing through the OpenLaszlo code.



This diagram highlights the key portions of a controller JSP page and a serialization JSP fragment. The get-data tag is used to construct and invoke a Get service to retrieve the list of nouns from the WebSphere Commerce Server. For each noun returned, a JSP fragment is included to convert it to the Management Center XML format. Within the JSP fragment, character data (CDATA) tags are used to wrap text values that are sent to Management Center.

If you customized the Management Center to add UserData into your noun, the serialization JSP fragments ensure that this user data is transformed into the Management Center object. Management Center objects that correspond to information in a UserData element must use the naming convention of xattr_*name*, where *name* is the name attribute of the value and the property value is the value. When the change or process URL request is initiated, the configured UserData element that is associated with xattr is populated appropriately. For example, the UserData noun might be xattr_WarrantyTerms. For more information on User Data, see the “Adding data to an existing Management Center object” topic in the Information Center.

Finally, additional JSP fragments are included to continue the serialization.



Continuing from the previous slide, this diagram demonstrates the mapping between a noun (top left) and a Management Center object (bottom right). The noun is in the BOD format returned by the WebSphere Commerce server from a get-data request and the Management Center object is the output of the serialization JSPs.

Creating a custom Get request

- Create a directory under WebContent/jsp to store your custom files
  - For example mycompany/catalog for a catalog extension
- Create a JSP fragment file named Serialize*ClientObject*.jspf
- Create a controller JSP page named Get*ClientObject*.jsp
  - Use `jsp:directive.include` to include the serialization fragment
- Update struts-extension.xml

In order to retrieve a custom Management Center object, you must add code to enable the Management Center Web application to mediate the request as described in the previous slides.

From within WebSphere Commerce Developer, navigate to the LOBTools Web application project. If one does not exist, create a directory under the WebContent/jsp directory that classifies the purpose of the Management Center object. For example, if the Management Center object is an extension of the catalog tool, create a directory structure called mycompany/catalog such that mycompany indicates your Management Center object extensions and catalog represents the catalog tool.

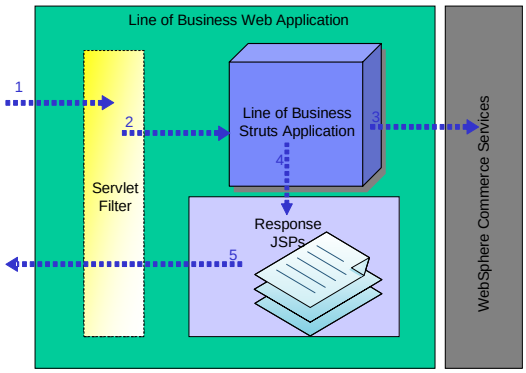
In the new directory, create a JSP fragment file named Serialize*ClientObject*.jspf where *ClientObject* is the name of your custom object. Implement the JSP fragment assuming a variable name representing the Management Center object has been specified and the JSP fragment that transforms the Management Center object to the Management Center object representation. In the same directory, create a controller JSP page to retrieve the Management Center object from the service. Use jsp:directive.include to use the serialization fragment to transform the object into the Management Center object representation. The name of the controller JSP page should match its usage. For example if the purpose of the request is to get the Warranty information of a Product, the name of the JSP should be GetProductChildren-Warranty.jsp.

Next, you need to define a new Struts forward action. Open the struts-extension.xml file and add a new forward action, which forwards to the newly created JSP page. The name of the Struts action should be the same as the name of the JSP page.

Restart the WebSphere Commerce server and test your changes.

Flow of a process request

1. The request enters the servlet filter to resolve the authentication information for the request.
2. The Struts action is executed. This builds the business object document request based on the given URL parameters.
3. The Struts action calls the client library to make the service request.
4. The Struts action processes the business object document response and forwards to the JSP to display the response.
5. The JSP formats the objects returned by the Struts action into the XML response Management Center expects.



The life cycle of a URL request to process (create or change) Management Center objects is shown here.

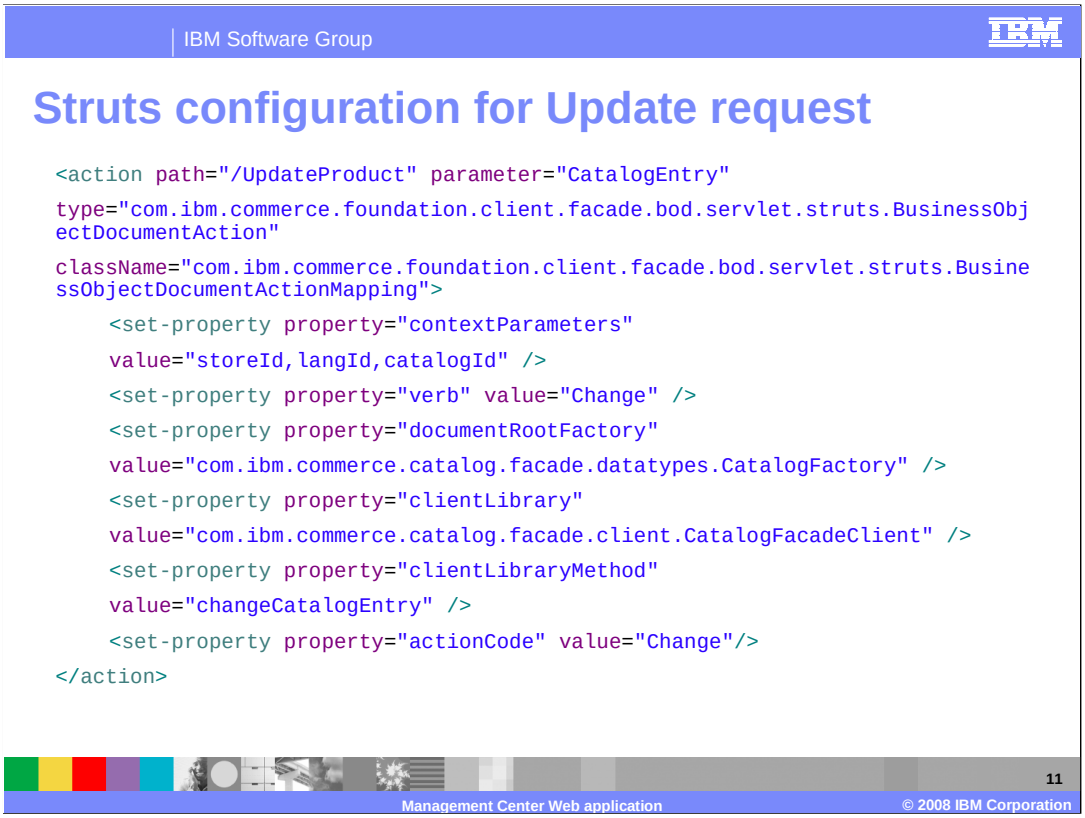
Step 1. The Management Center issues a URL request. The URL request enters the identity token servlet filter to resolve the authentication information for the request.

Step 2. The Struts action runs, building the appropriate change or process request that matches the specific URI invoked.

Step 3. The client library is used to issue the BOD request and receive the BOD response.

Step 4. The BOD response is parsed into a flattened Java™ map. The Java map and nouns in the response are forwarded to the Struts forward. Based on the result of the BOD request, the resulting Struts forward is one of success, failure, or error.

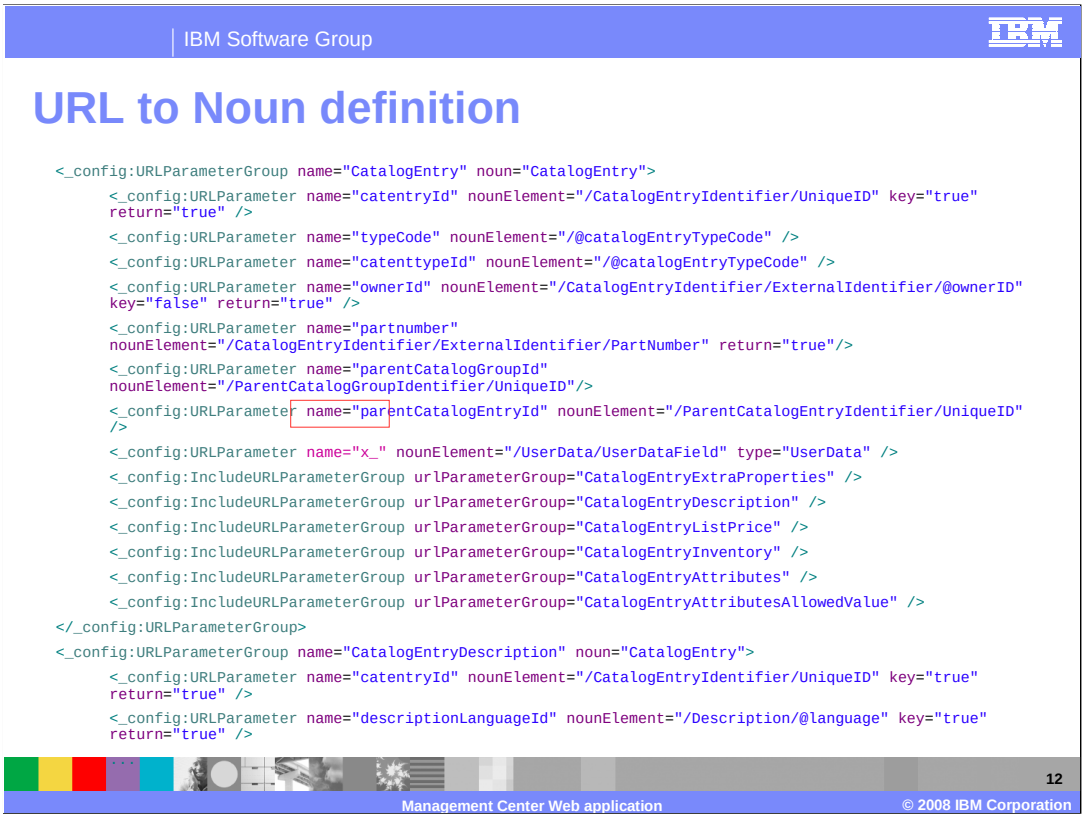
Step 5. The appropriate JSP page creates the response for the request. This response contains the information related to the result of the request, including any changed information that the Management Center requires as a result of the request.



Here is an example of the Struts configuration used to build an UpdateProduct service request. The URL follows the naming convention CreateObject, UpdateObject or DeleteObject. In this example the action is Update and the object is Product. The parameter attribute points to a definition in an external mapping file that defines how the URL name-value pairs are mapped to the CatalogEntry service definition. This external mapping file will be discussed in the next slide.

The type, BusinessObjectDocumentAction, is the generic Struts action that knows how to build Business Object Documents. Classname specifies the Struts ActionMapping class that defines the additional properties for this request.

The properties include contextParameters which defines the context parameters that are needed for this request and is included in the application area of the service request. Also included in the properties is the request verb. The documentRootFactory, clientLibrary and clientLibraryMethod properties define the Java artifacts that are used to create the object and invoke the service. Finally, the actionCode property specifies what operation the request is performing.



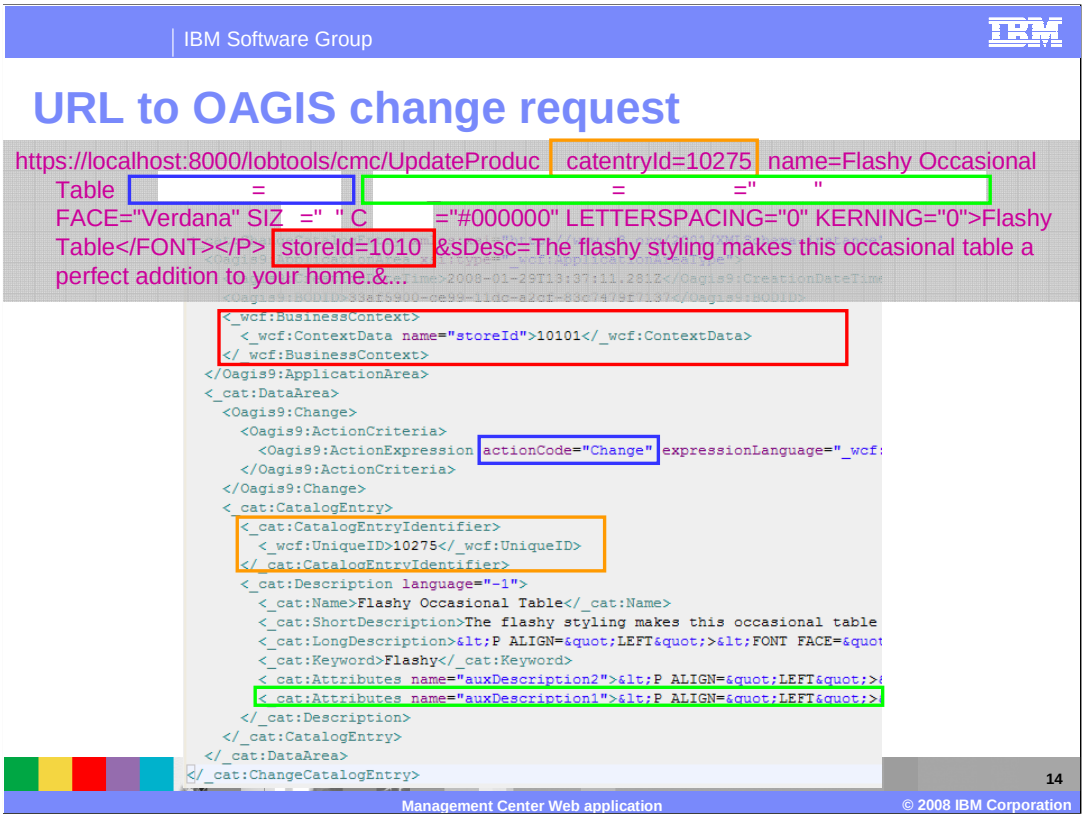
As mentioned in the previous slide, there is an external mapping file that maps the name of each Management Center URL parameter to the corresponding noun portion of the BOD. This file, known as a Management Center object definition file, contains all the information needed to convert a Management Center request into a BOD request. This slide looks at one section of the object definition file, the URL definition section.

When the Management Center performs an action, the properties of the Management Center object are passed as URL parameters that represent a noun element. The name of each URL parameter corresponds to the name and value of the associated Management Center object. In the Struts configuration file on the previous slide, the attribute named parameter has a value of CatalogEntry. Here, you see a URLParameterGroup called CatalogEntry that defines how each URL parameter maps to the CatalogEntry noun. The Management Center Web application iterates through request parameters and, for each parameter, determines if there is an element in the noun that is defined in the configuration file. For every parameter that has the corresponding definition or association, the noun element is populated with the URL parameter value. For example, the configuration shown has defined catalogEntryId to */CatalogEntry/CatalogEntryIdentifier/UniqueID.* If the URL request has catentryId=1234, then the noun produced has UniqueID populated with the value 1234.

Note the URLParameter mapping half way down with the name “x_”. This is used to map URL parameters into the UserData portion of the noun. User data must use parameters of the form x_*name*, where *name* is the name of the user data field.

Within the same configuration file is a noun definition section that identifies which parts of a noun can be modified by a Change or Process request. These are called a noun's changeable parts and are used to notify the server what portion of a noun has been modified.

To transform a Management Center URL to a Change BOD, you must identify the changeable parts of the noun. When a noun part is identified as changeable, an ActionExpression that points to that part of the noun is created and added to the Change verb. Each changeable part of the populated noun has an associated ActionExpression. The end result is a populated noun and a corresponding Change verb that contains all of the Change instructions for the Change request. Based on this configuration, any time that an element is populated, it is added to a set of XPaths that must be added for the change actions.



The color-coded mappings on this slide show how different URL parameters map into the application area, verb and noun of the OAGIS BOD.

The storeId is part of the application context for this request and is mapped into the ApplicationArea portion of the BOD.

To create the Change verb, a special URL parameter called actionCode is used to define the action code that must be part of the action expression. The URL to Change verb transformation has two key characteristics. First, the actionCode specified applies to all the parts of the associated noun. Although the Change request can change multiple parts and each action expression can have a different action code, this scheme assumes that the actionCode for the noun applies to all action expressions for that noun. This is a limitation of the mapping scheme of actionCode for building the Change service request. Second, the actionCode parameter value cannot be multi-valued. If multiple values are specified, only the first value is used. This characteristic is different than Process verbs, in which the actionCode can be multi-valued.

The noun portion of the BOD is populated based on the URL parameters and the noun and URL definitions described on the previous slide. In this example you can see the catentryId parameter being mapped to the noun's UniqueID element. This example also shows additional custom data, the xdesc_auxDescription1 parameter, being passed in the URL request. This is an example of UserData and is not part of the standard noun definition.

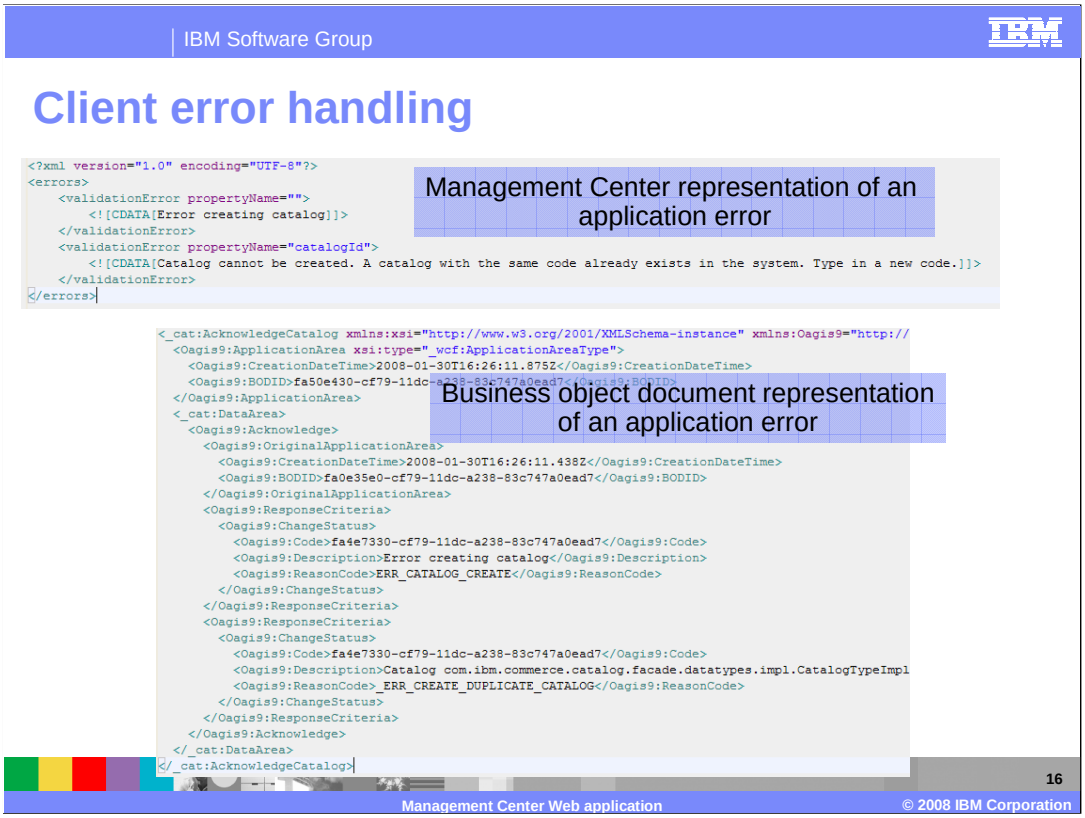
Creating a custom process request

- Create a mapping XML file in WebContent/WEB-INF/config
  - For example mycompany-catalog-clientobjects.xml for a catalog extension
  - Specify Noun
    - Include modifiable parts and UserData
  - Create URL mapping
  - Specify resource bundle that defines error messages
- Update struts-extension.xml
  - Add XML file
  - Create new BusinessObjectDocument Struts actions

You can define authoring services such as create, save, and delete for a custom Management Center object by adding your own Process request. To perform this customization you need to define the URL parameter mapping to the noun in the BOD request. You also need to configure the Struts action to indicate the verb and additional processing information. From within WebSphere Commerce Developer, navigate to the LOBTools Web application project. In the WebContent/WEB-INF/config directory, create an XML file that defines the mapping between elements of the noun and properties of the Management Center object. Use the name of the tool as part of the file name.

In the noun element, add the noun to the list of nouns. If this noun has any parts that can be modified, indicate those elements under the noun and specify the part attribute as true. If the noun has a UserData field that should be mapped to the Management Center object, the property name should begin with **x_.** In the URL element, add the mapping of the Management Center object to the noun. For each property of the Management Center object, specify the XPath location of the noun that the property represents. In the ValidationErrorMapping element, specify the resource bundle that is used to override the error messages returned by the server. A mapping of validation reason codes can be mapped to properties of each Management Center object. Open the struts-extension.xml file and add the new XML file to the list of files specified in the configuration property of the BusinessObjectDocumentPlugin. Create new BusinessObjectDocument Struts actions for each authoring service supported by the Management Center object. These Struts actions define the verb and additional information required to invoke the Business Object Document service.

When you are finished, save the configuration file and restart the WebSphere Commerce server to test your changes.



In the Management Center, a request might generate one of three types of errors. The first type of error occurs when the input to a client method cannot be converted into valid input for the request BOD. In this case, no request is sent to the WebSphere Commerce server. The second type of error occurs when a service is invoked and the request cannot be processed because of the information that has been specified in the request BOD. When this occurs, the response BOD contains error information found in the change status portion of the response verb. In the WebSphere Commerce OAGIS processing model, anytime the change status information is populated in the response BOD, the client can assume application errors have occurred and the change status contains the error information. The third type of errors is system exceptions. A system exception is an error condition which is outside the business tool domain. It probably means someone else needs to handle the error condition externally.

When one or more errors occur, the Management Center Web application creates an XML document that lists the validation errors or system errors. Each validation error contains the error message in the locale you specify in the Management Center client and optionally the associated parameter corresponding to the validation error. If no parameter is specified, the Management Center client assumes the message applies to the object that it sent as part of the request. This parameter helps the Management Center shell to identify a particular parameter that requires attention instead of just displaying an error. System errors contain a message you can pass on to the system administrator. The XML document is then returned as the response to the URL request.

Error reason mapping

```
<_config:ErrorDefinitions
primaryResourceBundle="com.ibm.commerce.catalog.client.lobtools.properties.CatalogLOBError
Messages"
alternateResourceBundle="extensions.com.ibm.commerce.catalog.client.lobtools.properties.Ca
talogLOBErrorMessages">
     <_config:ErrorGroup name="Catalog">
          <_config:ReasonCodeParameterAssociation
          reasonCode="_ERR_PARENT_CATENTRY_FOR_ITEM“ parameterName="parentCategoryId" />
          <_config:ReasonCodeParameterAssociation
          reasonCode="_ERR_CATALOG_MISSING_IDENTIFIER“ parameterName="identifier" />
          <_config:ReasonCodeParameterAssociation
          reasonCode="_ERR_CREATE_DUPLICATE_CATALOG“ parameterName="catalogId" />
          <_config:ReasonCodeParameterAssociation
          reasonCode="_ERR_CATALOG_IDENTIFIER_INV_LENGTH“ parameterName="catalogId" />
          <_config:ReasonCodeParameterAssociation
          reasonCode="_ERR_DELETE_OR_UPDATE_NON_EXIST_CATALOG“ parameterName="identifier"
          />
          ...
```

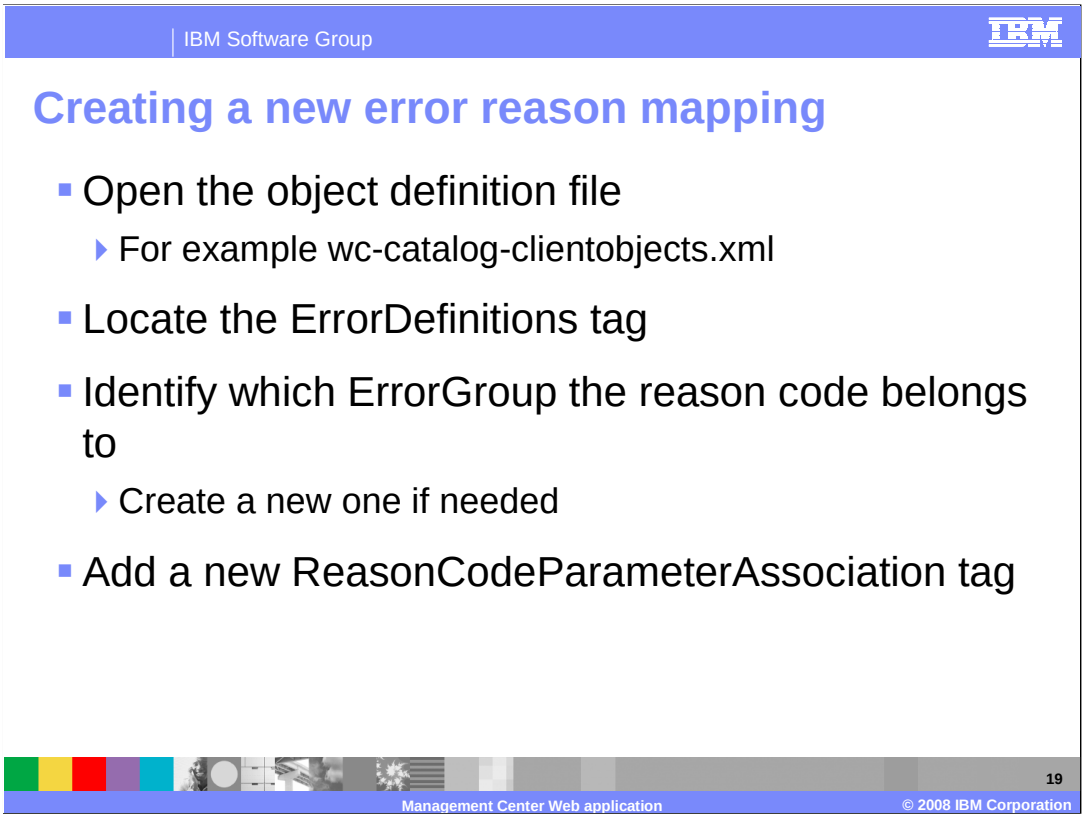

As described on the previous slide, if a request cannot be processed because of business logic validation errors, the BOD response contains error information in the change status element in the verb. The change status contains a reason code element that consists of a unique string representing the application error. In most cases, these unique reason codes correspond to a property of the Management Center object. Using the error definition section of the Management Center object definition file, each Management Center object can associate reason codes with a particular property. This mapping allows the error response to the Management Center to highlight the properties that caused the problem. A sample mapping is shown here.

Service error message override

```
...
CatalogGroup._APP_CATGROUP_PARENT_CATGROUP_DOES_NOT_EXISTS=Parent category does not
exist.
Catalog._ERR_PARENT_CATENTRY_FOR_ITEM=Parent catalog entry is missing. Type in a parent
catalog entry.
Catalog._ERR_CATALOG_MISSING_IDENTIFIER=Catalog code is missing. Type in a code.
Catalog._ERR_CREATE_DUPLICATE_CATALOG=Catalog cannot be created. A catalog with the same
code already exists in the system. Type in a new code.
Catalog._ERR_CATALOG_IDENTIFIER_INV_LENGTH=Catalog code is too long. The limit is 254
characters.
Catalog._ERR_DELETE_OR_UPDATE_NON_EXIST_CATALOG=Catalog cannot be changed or removed. It
does not exist in the system.
Catalog._ERR_CATALOG_DELETE_WITH_TOP_CATEGORY=Catalog cannot be removed. It has a top
category.
Catalog._ERR_CATALOG_DESC_MISSING_REQ_FIELDS=Catalog name is missing. Type in a new
name.
CatalogEntry._ERR_CATENTRY_DUPLICATE_PART_NUMBER_FOR_CATENTRY_CREATE=The specified code
is already in use. Type in a new catalog entry code.
CatalogEntry._ERR_CREATE_DUPLICATE_LIST_PRICE=The list price that you are trying to
create already exists in the system.
...
```


The error messages corresponding to the reason codes shown on the previous slide are defined in resource bundles. The resource bundle name is specified in the error definition section of the object definition file. Two resource bundles exist for each set of error code mappings. The primary resource bundle contains the predefined Management Center error messages and the alternate resource bundle is provided for you to change or extend the default error messages.

A sample resource bundle file is shown here. Notice that each error code is preceded by the name of the error group it is defined in. This allows you to specify different error messages for the same error code in different contexts.



You can add or change Management Center error messages by updating the object definition file described previously.

From within WebSphere Commerce Developer, navigate to the LOBTools Web application project. In the WebContent/WEB-INF/config directory, locate the file that describes the object with which you are working. Scroll down to the validation error mapping section which begins with the ErrorDefinitions tag. Identify which ErrorGroup contains the reason code mapping. If needed, create a new ErrorGroup for the Management Center object. Create the mapping by adding a new ReasonCodeParameterAssociation tag for the service reason code and the property of the Management Center object with which it should be associated.

When you are finished, save the configuration file and restart the WebSphere Commerce server to test your changes.

Adding a validation error message

- Open the ErrorMessages resource bundle
  - Located in src/extensions.com.ibm.commerce.*tool*.client.lobtools.properties
  - Create the file if it doesn't exist
- Add a new property of the format *ClientObject.REASON CODE*
- Specify the text message to be displayed

To add a validation message returned to the Management Center, add a new property in the alternate resource bundle defined in the object definition file.

From within WebSphere Commerce Developer, navigate to the LOBTools Web application project. In the src directory, locate the alternate resources file for the Management Center tool you are working with. If this is your first custom message, you need to create the extensions directory and resource bundle.

Add a property with the name *ClientObject.REASON CODE* where *ClientObject* is the name of the Management Center object and *REASON_CODE* is the code specified in the object definition file. Finally, add the text string for your new message.

When you are finished, save the resource bundle file and restart the WebSphere Commerce server to test your changes.

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_WCS6003_CMCMediationWebApp.ppt

This module is also available in PDF format at: ../WCS6003_CMCMediationWebApp.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:

WebSphere

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

Verdana is a registered trademarks of Microsoft Corporation in the United States, other countries, or both.

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

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.