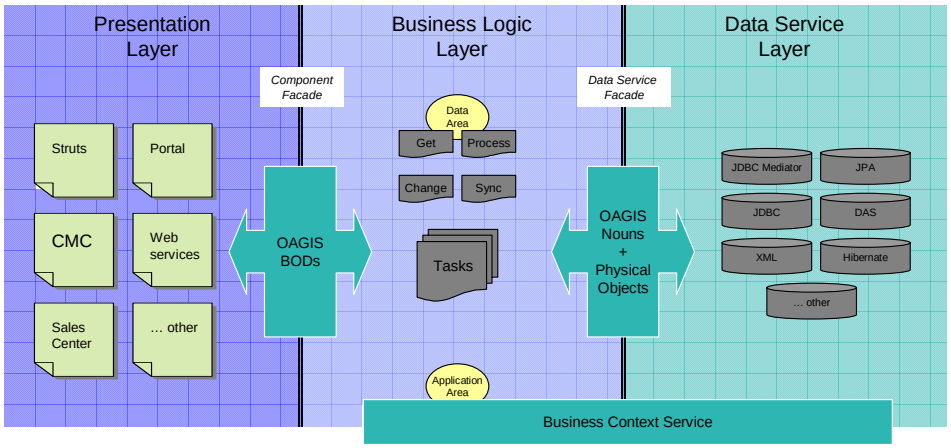
This presentation describes the OAGIS message architecture.

Agenda

- Message format
- Request messages
- Response messages

This presentation discusses OAGIS message format and the detailed structure of both request and response messages.

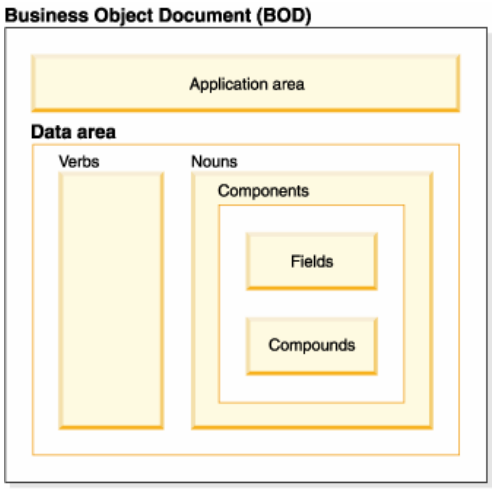
Architectural layers



The architecture of the application contains three distinct layers. OAGIS messages are used by the presentation and business logic layers to exchange data. Retrieving business data or executing any business logic must be done through the OAGIS defined services of a service module.

The remainder of this presentation discusses the format and structure of OAGIS messages. If you are already familiar with this architecture, you can skip this presentation and go to the presentation on Component Services, which discusses WebSphere Commerce specific architecture in more detail.

Business object document



- Application area
  - Contains application context information
- Verb
  - Get / Show
  - Process / Acknowledge
  - Change / Respond
  - Sync / ConfirmBOD
- Noun
  - Logical representation of the business object

The Business Object Document (BOD) is an open standard for a common horizontal message architecture developed by the Open Applications Group. BODs are business messages exchanged between software applications or components.

A BOD is a message structure that contains an application area and a data area to operate on. The application area of the BOD describes the application context of the request. In the context of a WebSphere Commerce service, the application area is where language and other session type data are stored. This information is common across all component facades and is used during processing.

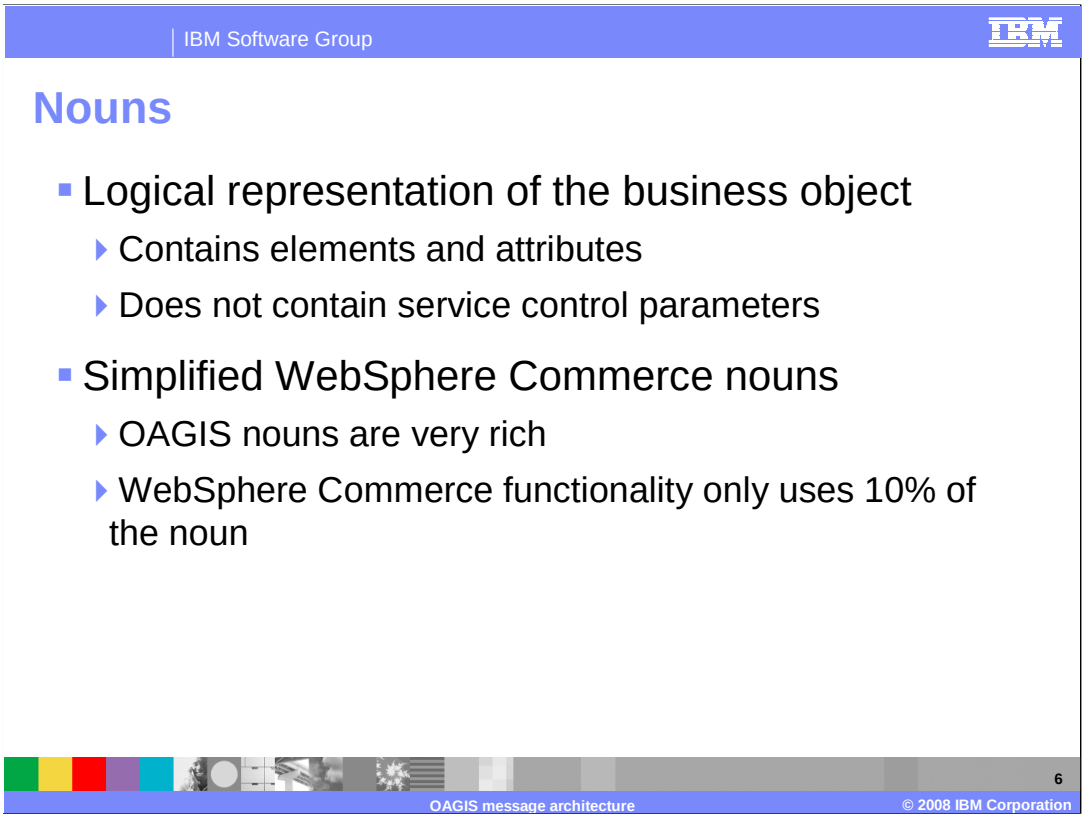
The data area contains the service operation, the verb, and the business object to operate on, the noun. The verb indicates what operation to perform or the response of the service operation. In the case of a request, it contains details of the operation and in the case of a response, it contains information pertaining to the response. The noun represents the data contained in the business object pertaining to the request or response.

Application area

- WebSphere Commerce specific session information
- Known context parameters
  - storeId, langId, catalogId
- Common session information independent of the business component
  - Input into the business context service
  - Managed by specific context implementations

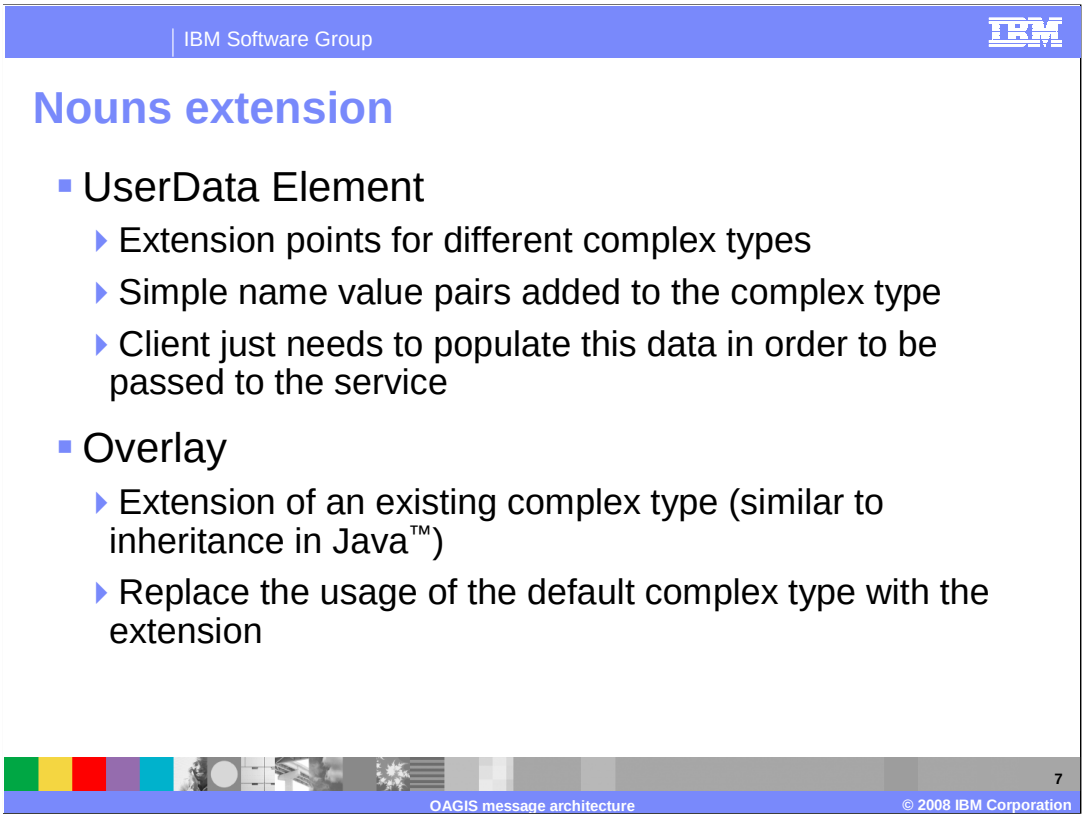
The purpose of the Application area is to provide a consistent method of passing common business context information into WebSphere Commerce. A number of context parameters are available including storeId, languageId and catalogId.

The business context service manages the contextual information used by each business component. The information is encapsulated within different types of business contexts; this process formalizes the context infrastructure and fosters reuse between different business models.



Any service that is fronted by an XML schema requires a logical model definition. For WebSphere Commerce, the logical model is represented as a noun. To reduce complexity, WebSphere Commerce uses its own simplified nouns, defined types, and primitive XML schema types, rather than using the nouns and base types provided by OAGIS.

Nouns define the name of each data element in the logical model, and assign that name to a data type. This data type can be a primitive XML schema type like boolean, or you can define a complex type. A complex type is a construct of data elements like CurrencyType which contains price (represented by a double type) and currency (represented as a string). WebSphere Commerce provides some predefined complex type constructs which are shared among all of the nouns.



There are two ways to add more data to an existing request and response document. One way is to take advantage of the UserData hooks within the existing types to pass unstructured information. The other way is to use the Overlay methodology.

Within the many complex types of nouns, UserData elements have been added as an easy way to extend the data that is passed from the client to the server. If you want to add information to the noun, or to the complex type of the noun, populate these UserData elements. As a result, without any coding effort on the server side or regeneration of the Web service, this data appears on the server side and can be processed immediately. The limitation of this method is UserData provides the ability to pass name-value pairs only.

Overlays, a part of the OAGIS model, extend complex types similar to the inheritance model in Java. Within the Overlay model, you extend your complex type defined within the noun and generate Java code to represent that complex type. More specifically, where you referenced the default WebSphere Commerce version of the complex type, you reference your new version of the complex type and substitute it in. Within the client server framework, when the complex type is sent from the client to the server the business logic retrieves your version of the extended complex type. This model provides another way to add additional information, as it allows you to create structured Java objects to better represent the data that you want to flow from one system to another.

OAGIS verbs

- An operation to be performed on a noun
  - Represent the request or response
  - Request verbs contain expressions which represent the action to perform
  - Response verbs indicate the result
    - Communicate Application Errors
  - Verbs are high level and can support new actions
  - Only recommending the use of eight verbs
    - Get, Show, Process, Acknowledge, Change, Respond, Sync and Confirm

A verb represents an operation that can be performed on a noun. For example, an "Order" noun has a specified operation that is performed in response to the verb "Get" (in this case, providing details to the client about the Order).

Each request verb has a corresponding response verb to return the result of the request. If a request cannot be processed, error information is returned 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.

Although OAGIS messaging defines many verbs WebSphere Commerce uses only the verbs shown here. Four are request verbs and four are response verbs. The request and response verbs work in pairs: Get and Show, Process and Acknowledge, Change and Respond, Sync and Confirm. These pairs are discussed in more detail in the coming slides.

Get request / Show response

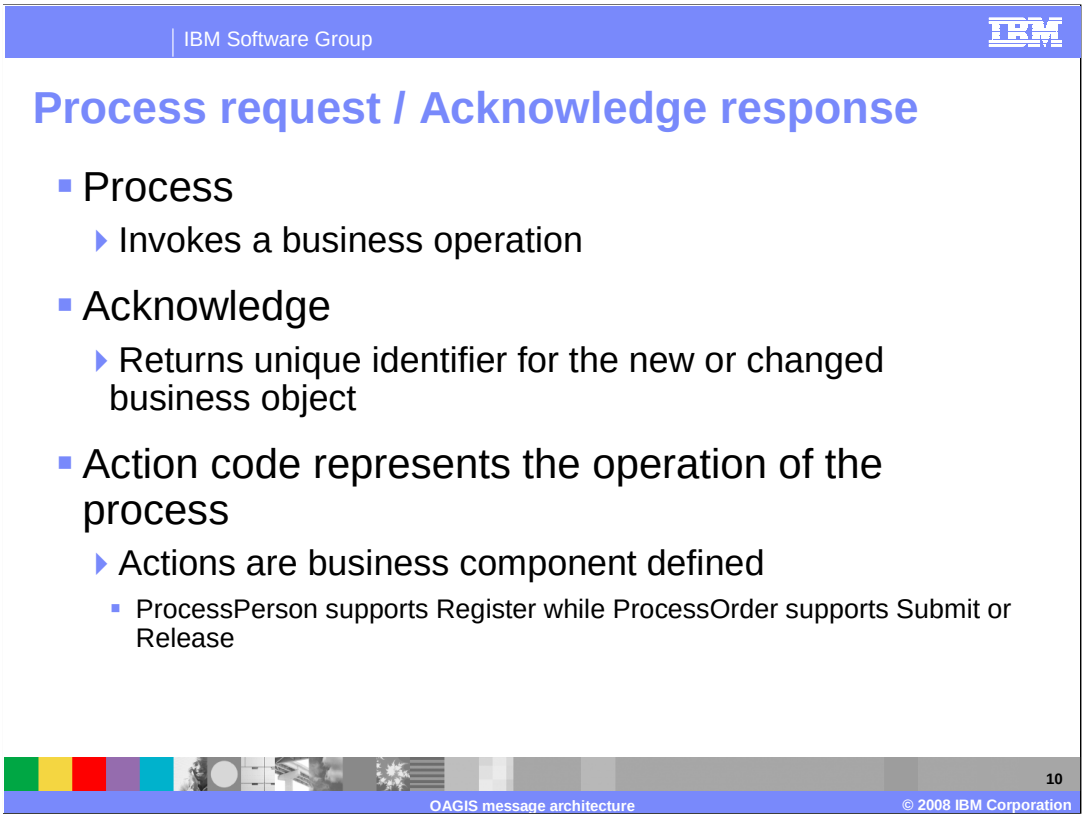
- Get
  - Request verb
  - Uses extended XPath notation to represent search clause
- Show
  - Response verb
  - Includes the list of nouns matching the expression
- Get request similar to a SQL statement
  - XPath expression = Where clause
  - Access Profile = columns to select
  - Noun = tables

Get requests are requests for data, and use the Get verb, and return a Show response. The Get verb indicates the search criteria used to retrieve the business objects along with paging options for paged requests. The Show response includes the business objects that match the search criteria.

Following the OAGIS model for read requests, the best practice is for every read request to be represented as a GetNoun document. In this case, Get is the action to perform and the noun is the business object or objects to return. As a response to the Get request, the Show document is returned with the requested data. The search expressions are XPath expressions which represent a query against the respective noun to indicate the data to be returned. As part of the extension to the XPath syntax, there is a way to pass control information to indicate the amount of data to return, ordering and other search result control information.

An access profile can be used to scope the response data. For example, to return a specific view of the data (such as a search view of a catalog entry), or to return the data in all languages for authoring purposes. As an extension to the XPath syntax, the access profile name-value pair should be added as a prefix to the XPath expression in curly brackets ({}). You must always specify the access profile.

There are cases when the results need to be paged so the service request returns only a subset of the data that matches the criteria. To control the number of records returned, the Get verb has paging attributes which can be used to indicate the number of records to return and where to start from on the list of information to return. The maxItems attribute is used to indicate the number of data elements returned from the query. The recordSetStartNumber attribute is the starting index of the record set to start returning information.



The Process request type is used when the client is submitting a new business object, canceling an existing business object, or invoking processing logic against an existing business object. A Process request is different from a Change request in that it invokes a business process that acts on the entire noun. Change requests can act on noun parts. The Acknowledge response includes the shell of the business object that was processed, containing at a minimum the unique identifier for the new or changed business object.

On Process requests, actions are user-defined. Example process actions include: Submit, Register, or Release. The action must indicate the business operation that needs to take place on the included noun. There is no predefined list of actions that can be associated with the Process verb, because the action is specific to the supported business process associated with the business object.

Change request / Respond response

- Change
  - Used to notify the master repository that it should change a business object it owns
  - Noun contains the information to uniquely identify itself and the attributes to change
- Respond
  - Returns unique identifier for the changed business object
- ActionExpression indicates the action to perform and an XPath pointer to the part of the noun to change
  - XPath can use the syntax to include control parameters
- Finite set of recommended change actions
  - Add, Change, or Delete

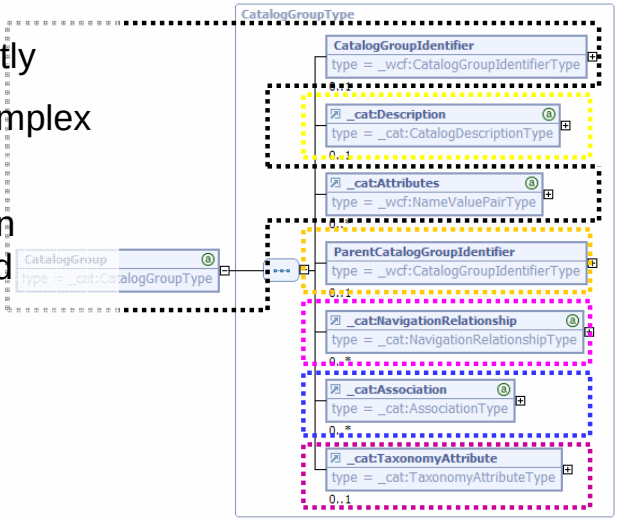
The Change request type is used to notify the master object repository that it should change a business object that it owns. The noun contains the information to uniquely identify itself and the attributes to change. For example, when adding a contact to a customer's contact list (address book), the noun specified contains the contact information to add along with the information to uniquely identify the Person business object. The response to a Change request is a Respond BOD that contains a Response verb and a noun. The noun refers to the data that was changed. The Respond verb contains the response criteria one for each action expression specified in the request.

The action expression uses XPath. The XPath expression represents a pointer within the noun to the business object to modify. Control parameters can be included as an extension to the XPath syntax.

Within the Change verb, there should only be one action expression where the actionCode attribute points out the operation to perform. The allowed operations for change are Add, Change and Delete.

Changeable parts

- Breaks down the noun into sections that can be changed independently
- Represented by a complex type
- The action expression points to the identified changeable parts



Change requests can operate on individual pieces of a noun called changeable parts. By identifying which parts of a noun have been modified, the client can provide the server with the information it needs to efficiently process the request.

When an element or attribute is set for an identified changeable part, 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.

Sync request / ConfirmBOD response

- Sync
  - Used to notify interested parties of the current state of the business objects that the system manages
  - System that contains the master data record initiates the Sync request
- ConfirmBOD
  - Generic result to indicate the result of processing the Sync request

The Sync request type is used to notify interested parties about the current state of the business objects that the system manages. Only the system that contains the master data record should be sending Sync requests. Sync is used to synchronize the business object or objects across systems and is initiated by the system that holds the master data record for the business object.

A common example of when sync is used is for asynchronous updates by the Java Message Service. Sync is not limited to this usage. When backend systems are pushing updates, these updates are sent asynchronously and reliably. For example, the master data record for member services uses Sync BOD to broadcast updates. If the SAP backend is the master data record for Order, then a Sync BOD request is received when the order status changes.

As a response to a Sync request, the ConfirmBOD response is used to indicate whether processing was successful. The confirm response contains the success or failure message as a result of the synchronization. The response does not include any information about the noun itself. The purpose of Sync is to synchronize the system so the nouns currently in the system match the nouns within the request message.

For Sync requests, actions are user-defined. The action must indicate the business operation that needs to take place on the included noun. There is no predefined list of actions that can be associated with the Sync verb.

This concludes the overview of OAGIS message processing.

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_MessageFormat.ppt

This module is also available in PDF format at: ../WCS6003_MessageFormat.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

Java, JDBC, 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.