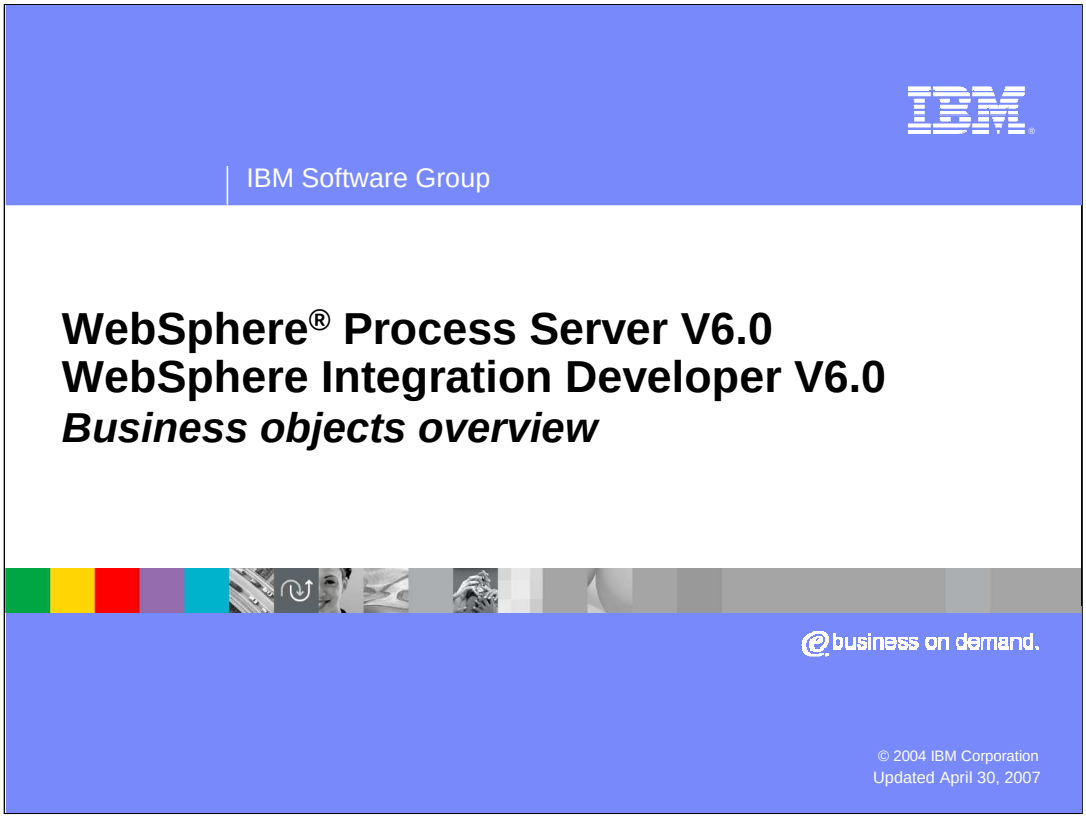
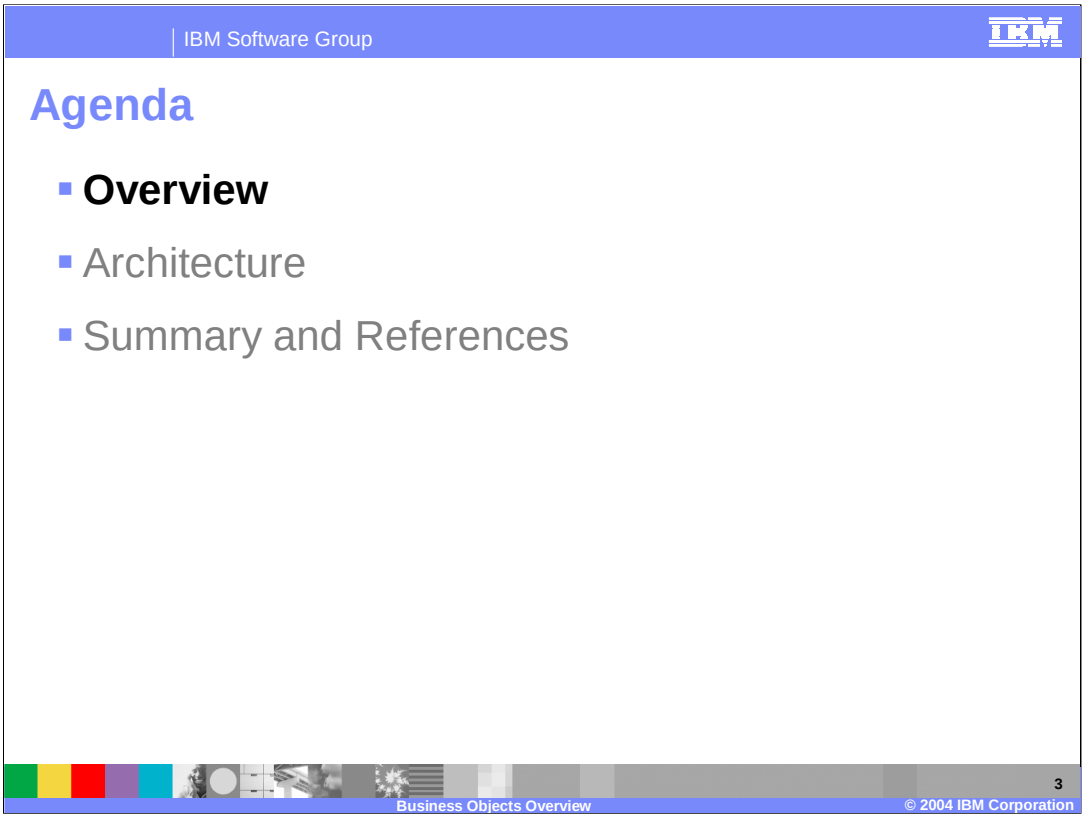


This presentation will provide an overview of business objects.

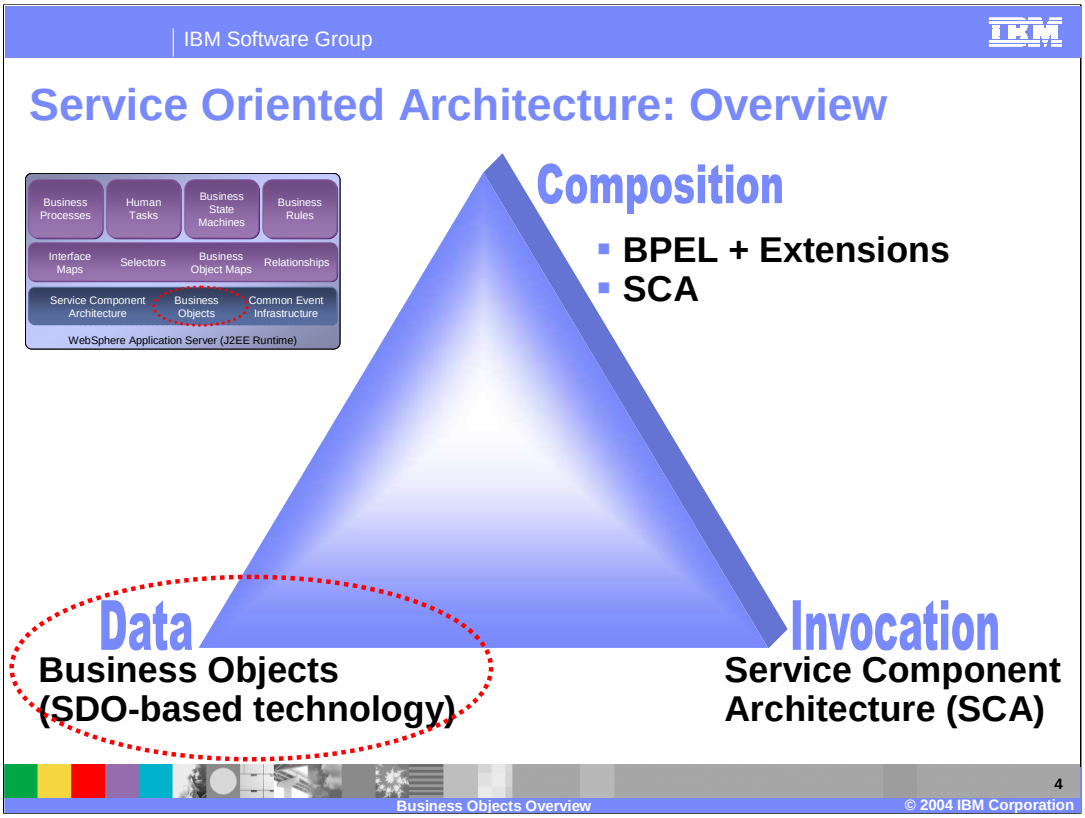
Goals

- Provide a brief overview of Service Data Objects (SDO)
- Introduce the Business Object framework

The goals of this presentation are to provide a brief overview of Service Data Objects and an introduction to the business object framework in WebSphere Process Server V6.



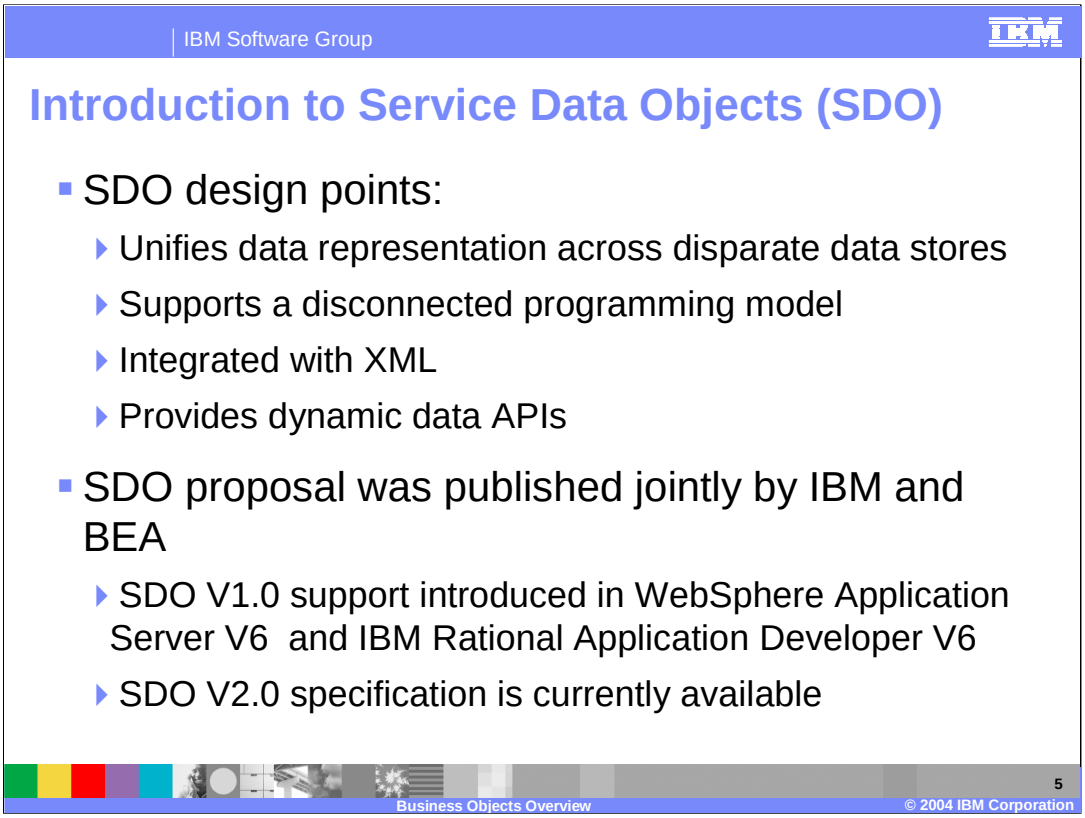
This section will provide an overview of business objects.



The triangle of truth is a simple way to look at the important architectural constructs that make up a service oriented architecture. When you consider what is needed to build a service oriented architecture, the triad that makes up the triangle of truth quickly emerges. Specifically, there must be a way to represent the data that is exchanged between services, a mechanism for invoking services, and a way to compose services into a larger integrated business application.

Today there are many different programming models for supporting each construct in the triangle of truth. This situation presents developers with the challenge of not only solving a particular business problem, but also choosing and understanding the appropriate implementation technology. One of the important goals of the WebSphere Process Server V6 SOA solution is to mitigate these complexities by converging the various programming models used for implementing service oriented business applications into a simplified programming model.

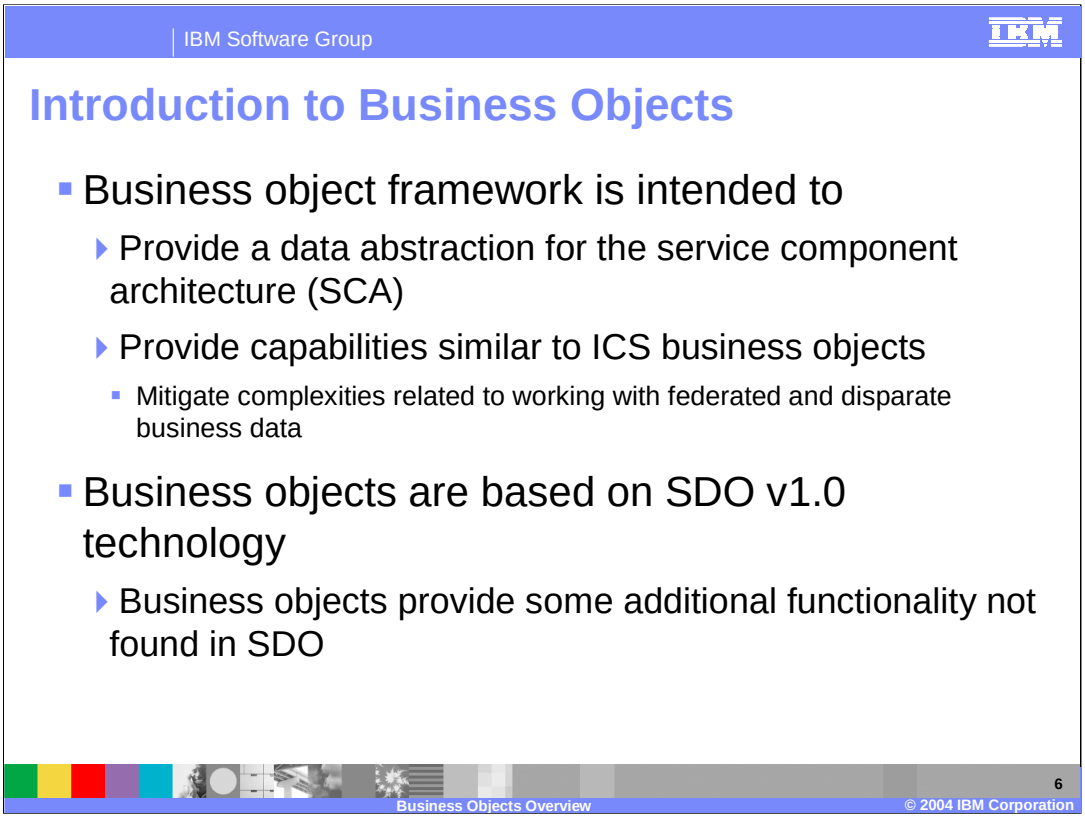
This presentation focuses specifically on the programming model used in WebSphere Process Server V6 for representing data that is exchanged between business components. Business data that is exchanged in an integrated application in WebSphere Process Server V6 is represented by constructs called business objects. Business objects are based upon a new data access technology called Service Data Objects (SDO).



Before discussing the specifics of the business object architecture, it is important to understand the architecture upon which business objects are based. Until now, developers have been faced with a wide range of data models and data access APIs to represent and retrieve data from a variety of backend data sources. For example, it is not uncommon for an enterprise application developer to need to know several data access technologies such as: JDBC, XML, JMS, Web Services, and Enterprise Information Systems. Unfortunately, this situation requires developers to become experts in many different data access technologies. The goal of SDO is to provide a programming model that will unify data representation across heterogeneous data sources and simplify application development for developers and tools providers. SDO provides this by offering a common API that can be used regardless of the backend data store being accessed. This common way of representing data also makes SDO an ideal choice for data abstraction in a service oriented architecture.

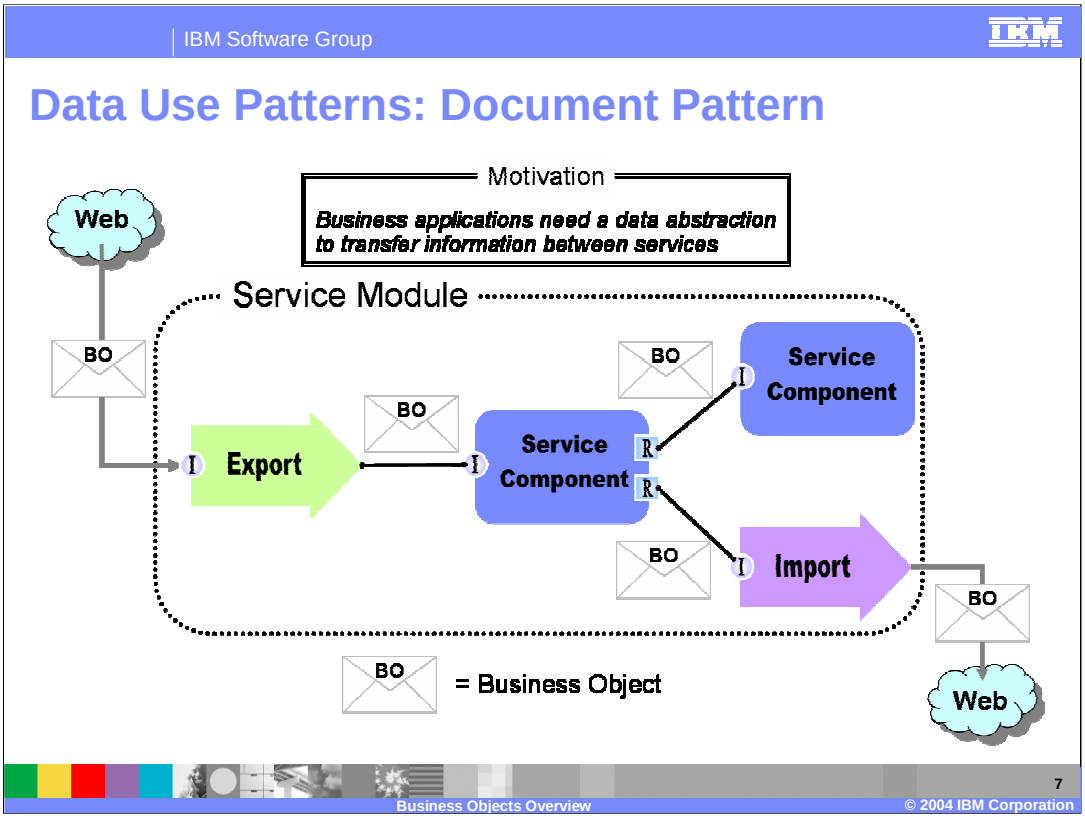
In addition to providing a programming model that unifies data access, there are several other key design features to note about SDO. First, support for some common programming patterns is built into the SDO architecture. Most significantly, SDO supports a disconnected programming model. Typically with this type of pattern a client may be disconnect from a particular data access service (DAS) while working with a set business data. However, when the client has completed processing and needs to apply changes to a backend data store by way of a DAS, a change summary is necessary in order to provide the appropriate level of data concurrency control. This change summary information has been built into the SDO programming model to support this common data access scenario. Another important design point to note is that SDO integrates well with XML. As a result, SDO naturally fits in with distributed service oriented applications where XML plays a very important role. Finally, SDO has been designed to support both dynamic and strongly typed data access APIs. However, in this this release only dynamic APIs are supported. The dynamic APIs are provided with the SDO object model and provide an interface that allows developers to access data even when the schema of the data is not known until runtime.

SDO is a relatively new programming model that has been proposed jointly by IBM and BEA. IBM's support for the initial v1.0 specification was introduced in the WebSphere Application Server V6 and IBM Rational Application Developer V6 products. Since that time, SDO architects have published a revised specification with the release of SDO V2.0.



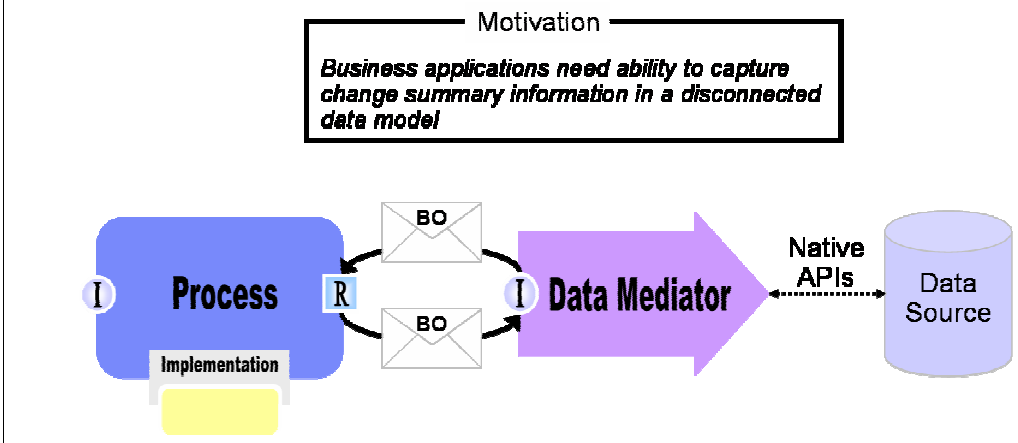
Business objects play an important role in the WebSphere Process Server V6 SOA solution as the data abstraction. This is an important goal of the business object framework, but in addition to this, the business object framework also provides some other important functions. Specifically, the business object framework was developed to provide functional capabilities similar to the business object construct found in WebSphere Interchange Server (ICS). The set of capabilities that have been adopted to support ICS style business object functions are needed to provide a way to help developers mitigate the complexities related to developing applications that work with federated and disparate business data as is commonly the case with integrated enterprise applications.

The business object framework in WebSphere Process Server V6 is based on SDO v1.0 technology. For this reason, an understanding of the SDO specification is beneficial when learning about the business object framework. Knowledge of the programming interfaces is also needed to begin writing client code that accesses business object data. Because the SDO specification is a relatively new technology, there are some areas where the business object framework has added some additional functionality not found in the v1.0 specification. This functionality has been added to version 2.0 of the SDO specification. In addition to this, some features have been added on top of SDO to provide the enhanced features needed to provide the ICS style business object capabilities. The additional features that are provided and are not part of the core SDO specification will be identified in this presentation.

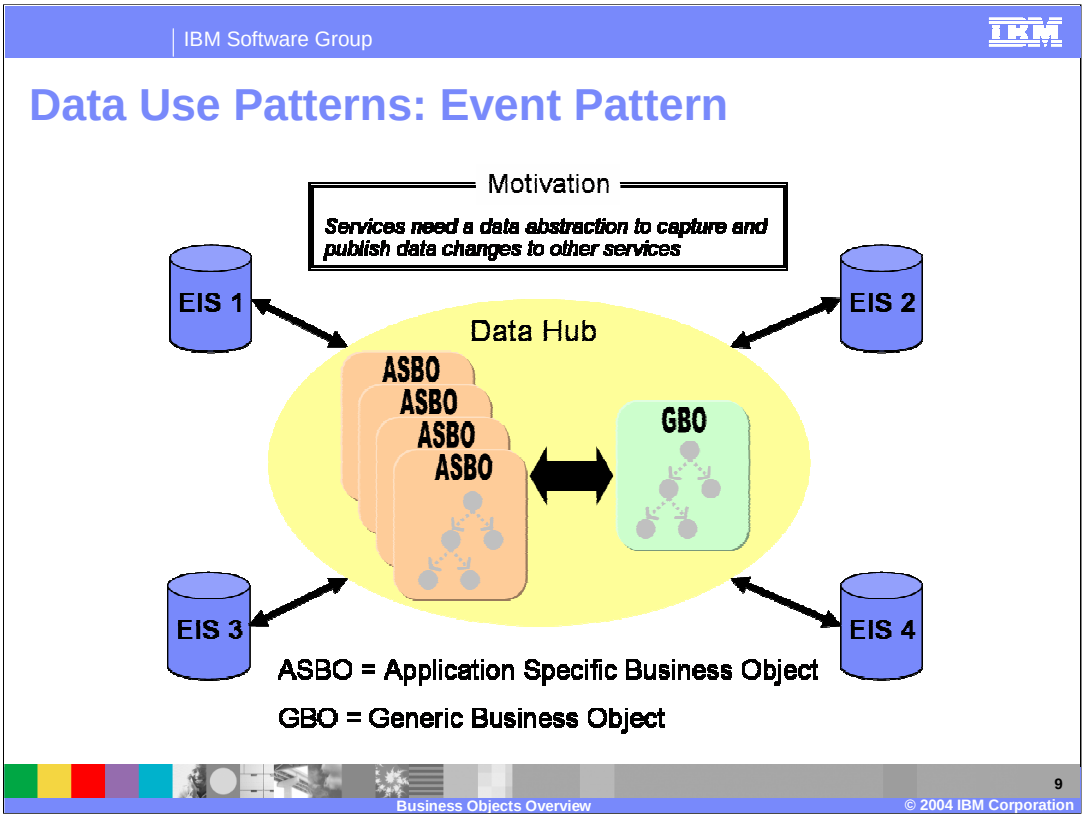


This diagram highlights one of the goals of the business object framework introduced on the previous slide. Specifically, business objects play an important role in providing the data abstraction for the service component architecture (SCA). Both SCA and SDO, which is the basis of business objects, have been designed to be complimentary service oriented technologies. This diagram illustrates how SCA provides the framework to define service components and compose these services into an integrated application. It also shows that business objects represent the data that flows between each service. Whether the interface associated with a particular service component is defined as a Java interface or a WSDL port type, the input and output parameters are represented using business objects.

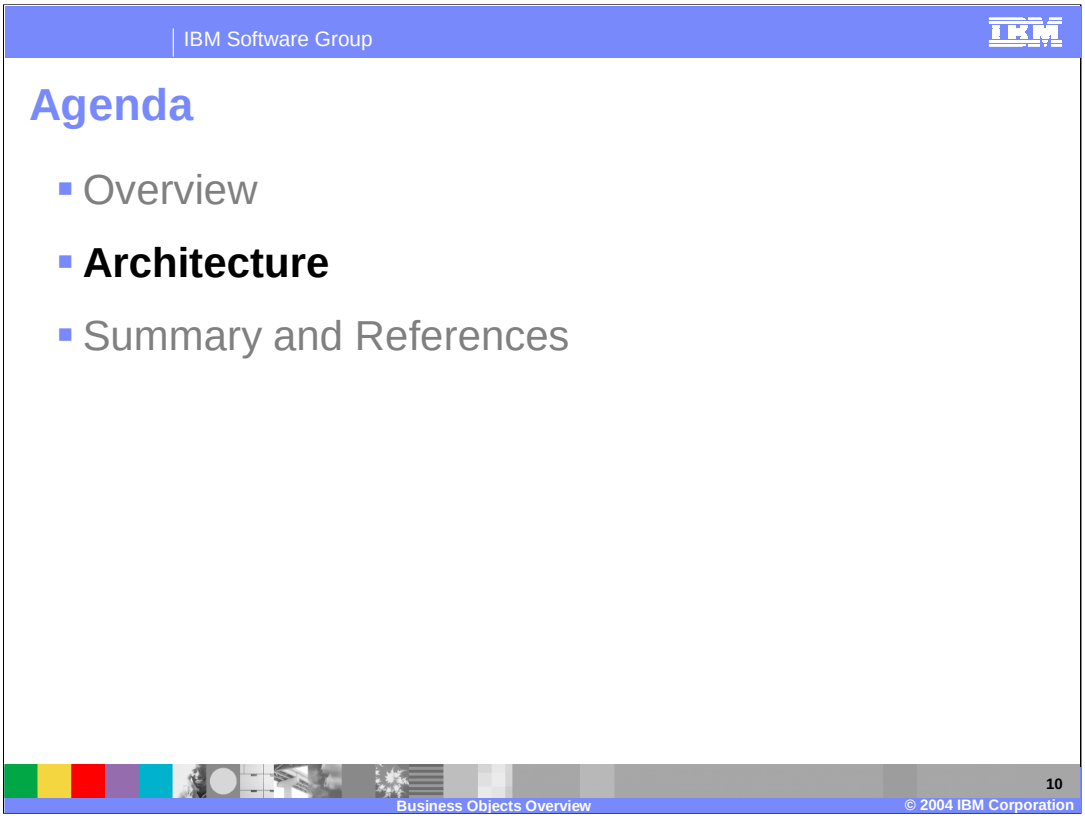
Data Use Patterns: Disconnected Data Pattern



The diagram on this slide highlights the disconnected data pattern. The motivation for this pattern is to provide support for business applications that require the ability to capture change summary information in a disconnected data model.



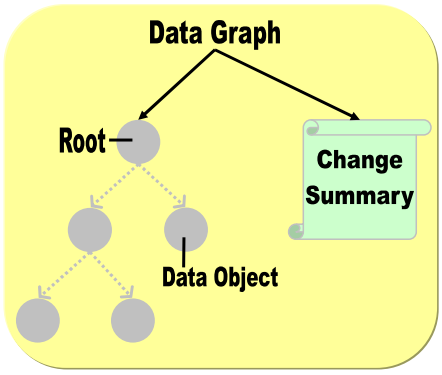
This diagram illustrates one of the other important goals of the business object framework that was introduced earlier. Specifically, business objects are needed to mitigate the complexities of working with disparate data systems. This diagram shows several EIS systems, each with a unique and application specific schema for representing the business data in the system. The complexity of this situation is that most enterprises will ultimately need a single view of the business data in their enterprise when building business processes and applications that are agnostic to the type of EIS system. To solve this problem, WebSphere Process Server V6 provides an architecture built upon business objects that allows mapping data from application specific to global business objects. This mapping infrastructure is central to the process of transferring information between disparate applications. Although this presentation does not provide the details regarding creation and use of data maps, it is important to introduce this concept here to understand how business objects fit into the overall WebSphere Process Server architecture.



This section will provide an overview of the business object architecture.

Service Data Objects Programming Model

- Data Object
  - Fundamental component in the SDO architecture
    - `commonj.sdo.DataObject`
  - Holds primitive data or multi-valued fields
  - Includes a reference to metadata
- Data Graph
  - Contains a root DataObject
  - Contains change summary information



Before discussing the details of the business object architecture, it is useful to understand a few key SDO concepts that will help you understand how business objects are designed and utilized in WebSphere Process Server V6. These concepts will help you understand exactly how business objects relate to SDO.

The fundamental concept in the SDO architecture is the data object. In fact, it is not uncommon to see the term SDO used interchangeably with the term data object. A data object is a data structure that holds primitive data and multi-valued fields, or other data objects. The data object also holds references to metadata that provides information about the data included in the data object (see the SDO Type and Properties interfaces). In the SDO programming model, data objects are represented by the commonj.sdo.DataObject java interface definition. This interface includes method definitions that allow clients to get and set the properties associated with the DataObject. As an example, consider modeling customer data with a SDO data object. The properties associated with the customer might be things like, firstName(String), lastName(String), customerID(long). The following is a sample of how you would use the DataObject API to get and set properties for the customer data object:

```
DataObject customer = …
customer.setString("firstName","John");
customer.setString("lastName","Doe");
customer.setInt("customerID", 123);
int id = customer.getInt(“customerID”);
```

Another important concept in the SDO architecture is the data graph. A data graph is a structure that encapsulates a set of data objects. From the top level data object contained in the graph, all other data objects are reachable by traversing the references from the root data object. Another important feature included in the data graph is a change summary that is used to log information about what data objects in the graph have changed during processing. In the SDO programming model, data graphs are represented by the commonj.sdo.DataGraph Java interface definition. In addition to this, the change summary information is defined by the commonj.sdo.ChangeSummary interface. A complete object model for SDO v1.0 is included in the Appendix of this presentation.

Business Object Framework: Key Concepts

Concept	Description
Business Object (BO)	Fundamental data structure for representing business data
Business Graph (BG)	Wrapper for a business object or hierarchy of business objects to provide enhanced information such as ▪ Change summary ▪ Event summary ▪ Verb
Business Object Type Metadata	Metadata provides the ability to annotate business objects with application specific information
Business Object Services	A set of services provided to facilitate working with business objects. These services are available in addition to the capabilities already provided by SDO v1.0

The table on this slide introduces several key concepts that make up the business object framework. The intent is to provide a high level view of the major pieces composing the entire framework before considering the details of each concept.

The business object is the fundamental data structure used to represent business data. It is not uncommon to hear the term business object used to refer to the entire framework, but in this presentation, the term business object is used to refer to the fundamental data structure for representing business data and not to the overall architecture. When discussing the overall architecture, the term 'business object framework' will be used. The business object is directly related to the SDO data object concept discussed previously. In fact, business objects in WebSphere Process Server V6 are represented in memory with the SDO type commonj.sdo.DataObject.

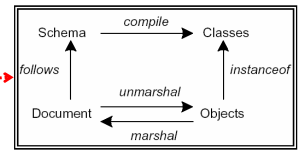
The business graph is used in the business object framework to wrap a top level business object and provide additional information used to enhance the data included the graph. In particular, the data graph includes the change summary for the data in the graph (similar to the SDO change summary information), event summary, and verb information used for data synchronization between EIS systems. The business graph is very similar to the concept of the data graph in the SDO architecture. However, notice that the event summary and the verb portion of the enhanced information are not included with the SDO data graph concept. The significance of the information included with the business graph will be discussed later in this presentation.

The business object type metadata provides the ability to annotate business objects with application specific information.

Business object services facilitate working with business objects. These services are available in addition to the capabilities already provided by SDO v1.0 and are needed to provide functionality where the initial version of SDO did not specify a solution. In the revised SDO specification these types of services have been included where applicable.

Business Object: Introduction

- Primary data structure for representing
  - Business data
  - Document literal message definitions
    - Example: Data types defined in WSDL definitions
- Represented in memory as an SDO commonj.sdo.DataObject
- Modeled using XML Schema
  - Supports importing XML Schema definitions
  - Supports the most commonly used XSD modeling techniques

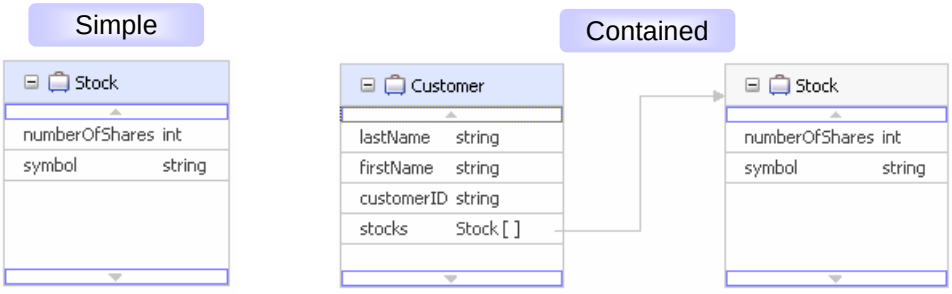


A business object is the primary structure for representing business data in the WebSphere Process Server V6 runtime. This also includes document literal message definitions as would be found in a WSDL definition when defining included data types. In relating business objects to SDO, the business object relates to the DataObject construct in SDO. In fact, business objects are represented in memory as an SDO commonj.sdo.DataObject. Therefore, if you will be doing development work that involves programmatically working with business objects, it is important to become familiar with the DataObject APIs. It is also important to point out that you might hear or see a business object referred to as a an SDO. This is because the SDO DataObject is used to represent a business object in the client programming model.

Currently, the only model for modeling or defining business objects is XML Schema. Because the business object framework supports the most commonly used XSD modeling techniques, business object definitions created by third party systems can be successfully imported and used in a WebSphere Process Server V6 application module. Development teams that have business objects defined using a model other than XML Schema must covert these business objects to XML Schema to use these definitions.

Business Object: Introduction (cont.)

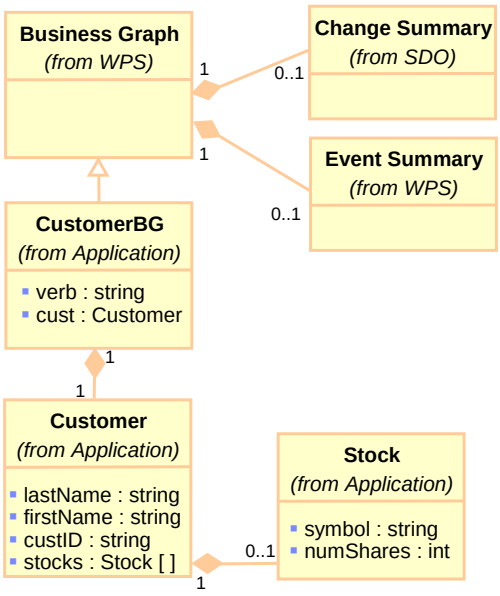
- Supports two types of business objects
  - Simple business objects
    - Composed only of scalar properties
  - Contained business objects
    - Composed of attributes that reference nested business object definitions



There are two types of business object definitions to be aware of. In the simplest case a business object can be constructed of only scalar properties as shown in the Stock business object on this slide. In addition to this, business objects can be defined to contain other business objects. In this case the business object is composed of one or more properties that reference a nested business object definition. The Customer business object shown on this slide has been included to illustrate an example of a contained business object. In this example, each customer is associated with a collection of stocks. The data type for the stocks property is the Stock business object.

Business Graph: Introduction

- Container around a top level BO
- Provides the following enriched capabilities related to the contained BO
  - Change Summary Header
  - Event Summary Header
  - Verb
- Use cases that illustrate fundamental capabilities
  - **Disconnected Data Pattern**: Captures changes to contained BO between processes
  - **Event Pattern**: Provides ability of EIS systems to capture and publish data changes to the business integration runtime



A business graph is a concept related to the SDO data graph concept. In the WebSphere Process Server V6 runtime, a business graph is used as a container to wrap a top level business object and provides several important capabilities that are related to the contained business object hierarchy. Before looking at the additional capabilities provided by the business graph, it is helpful to first consider the fundamental use cases for a business graph to learn why this additional information included with a business graph is useful.

The first use case to consider is a disconnected use model where there is a need to be able to capture changes made to the contained business object between processes. For example, consider a scenario where a service (service1) constructs a business graph based upon a query to a back end data store, and then passes that business graph to another service (service2) for processing. If the second service makes changes to the business objects contained in the business graph, and then subsequently calls the first service to apply those changes to the backend data store, then change summary information is necessary to assure the appropriate level of data concurrency.

Another important use case to consider is the situation where the information contained with the business graph is used to capture and publish data changes to the business integration runtime from an EIS system. In this case, the change summary, event summary, and verb associated with the business graph are all used to communicate the appropriate information needed to perform data synchronization among different EIS systems.

Consider the case where a Customer business object needs the enhancements provided by a business graph. For this, you would create a business graph called CustomerBG. The CustomerBG is defined using XML Schema and can be thought of as a specialized type of business object that includes the verb for the business graph and a single element of type Customer. In addition to this, the CustomerBG extends the Business Graph complex type provided by the WebSphere Process Server V6 business object framework runtime. The Business Graph complex type is the part of the business graph that provides the change summary, and event summary to a business graph.

Some final notes on business graphs to be aware of:

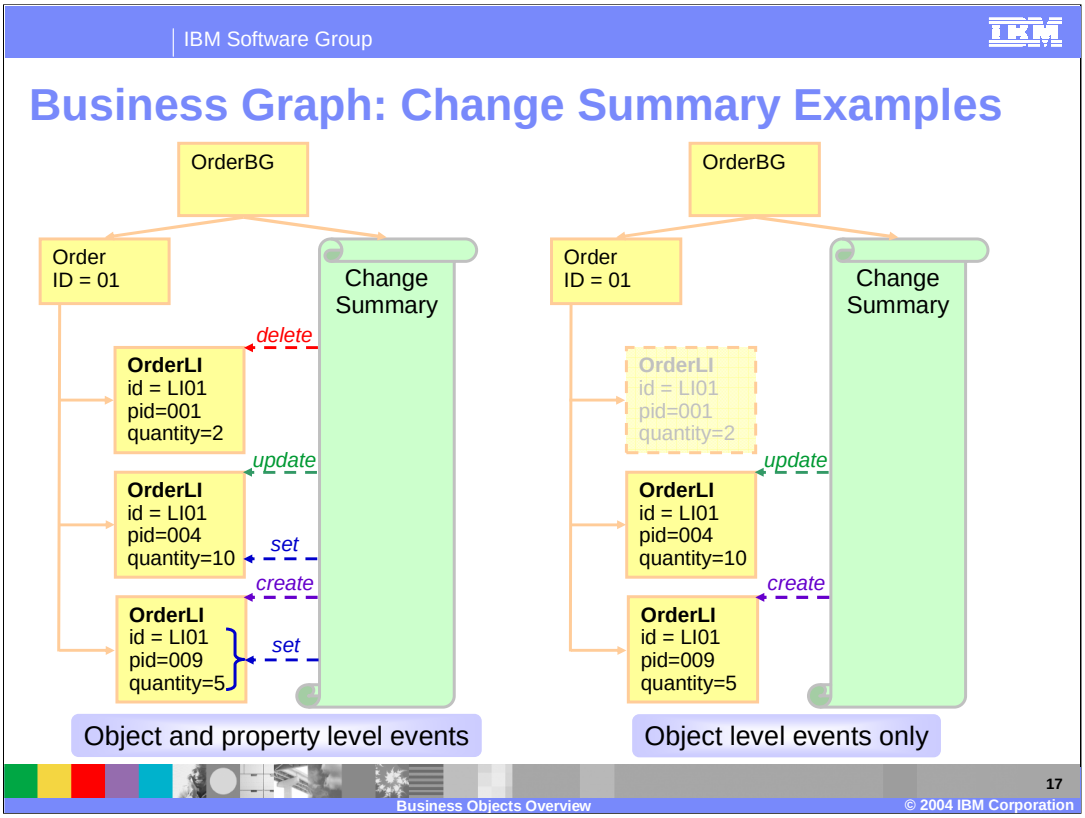
--> Business graphs are used for enrichment of only top level business objects. However, this does not mean that a contained business object in one scenario would not be a top level business object in another scenario and a candidate for enrichment with a business graph.

--> A business graph is represented by a commonj.sdo.DataObject in memory, but the event header and the change summary header can not be accessed using the normal DataObject APIs used to access contained DataObject properties.

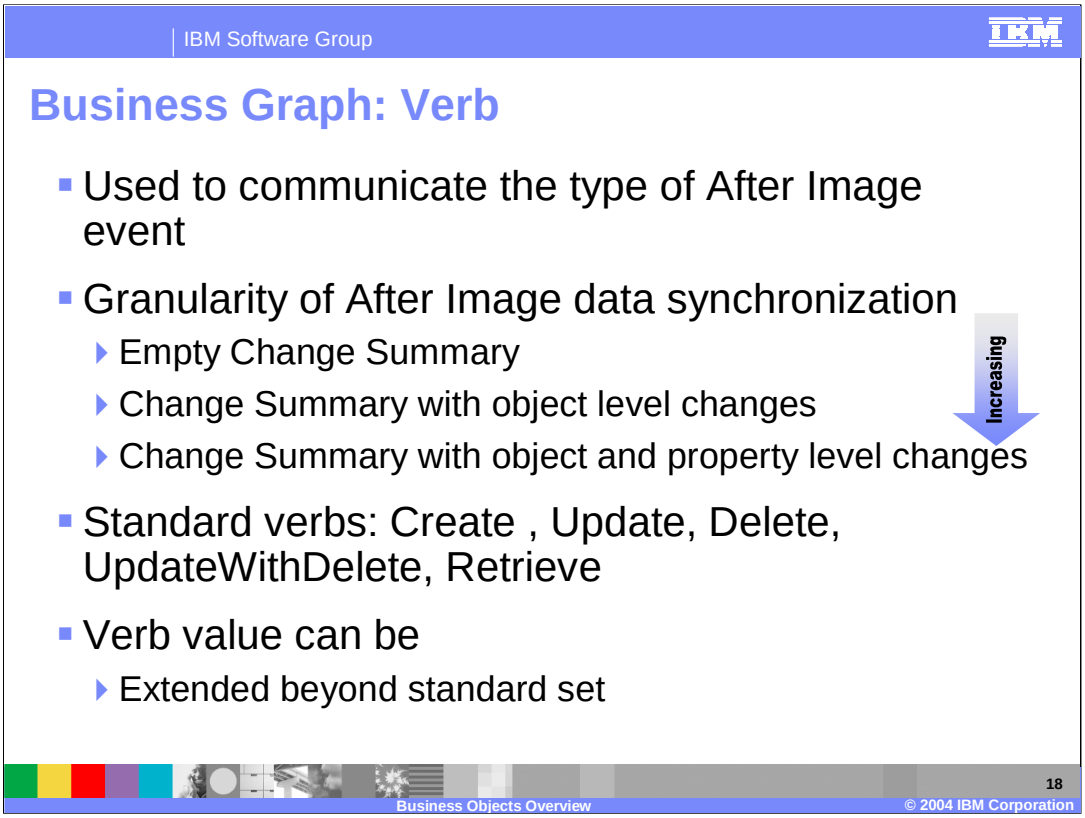
Business Graph: Change Summary

- Used to record changes to contained business objects in the business graph
- Used for both Disconnected Data Pattern and Event Pattern use models
- Two types of change summary usage
  - Implicit
    - Change logging is turned on and any changes to contained BOs are logged
    - Use commonj.sdo.ChangeSummary APIs
  - Explicit
    - Provides the ability to modify the change summary explicitly
    - Use com.ibm.bo.BOChangeSummary APIs

The change summary associated with a business graph is used to record changes made to the contained business object hierarchy wrapped by the business graph. This information is useful for both disconnected data pattern usage and command event use models (data synchronization between EIS systems). There are two ways to effect the change summary associated with a business graph. The first way is to make implicit change summary updates. This is done by turning on change summary logging through the commonj.sdo.ChangeSummary interface. Doing this will begin tracking changes made whenever any of the DataObject APIs are called for the business objects contained in the business graph. For more information on this type of change summary tracking, refer to the commonj.sdo.ChangeSummary interface. The other type of change summary tracking can be made explicitly by using a business object service that allows the change summary to be updated directly. This must be offered as a business object service because this capability is not available with the base SDO ChangeSummary functionality and is necessary in order to support the command event model.



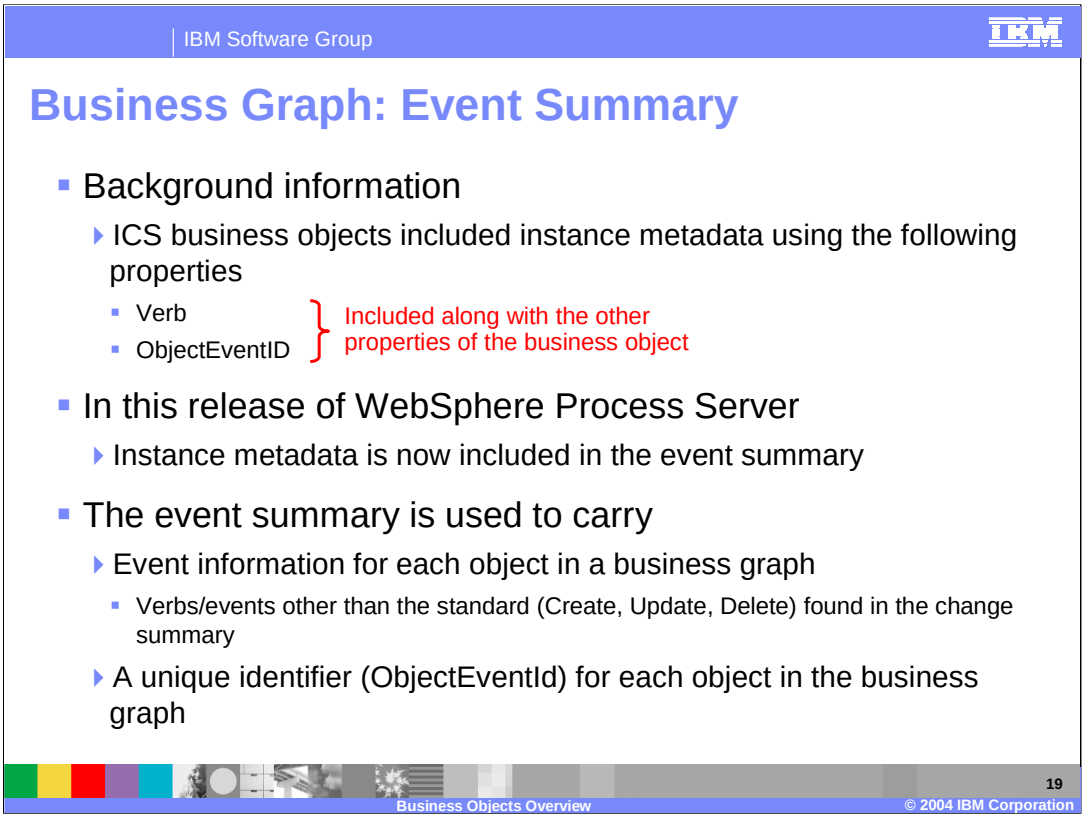
This slide provides an overview of the change summary and the level of granularity at which changes can be tracked. Shown on the left is a data graph that has captured both object and property level changes made to the contained business objects. On the right the change summary captures changes to object level events, and no property level changes.



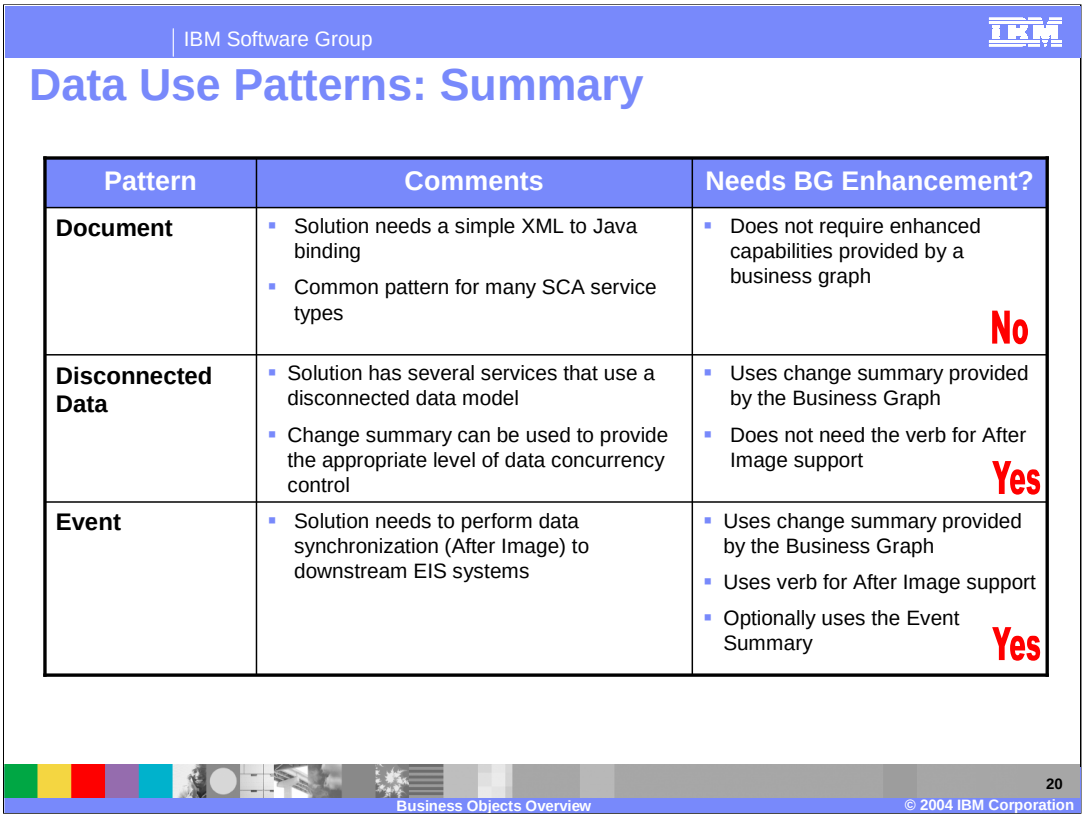
The verb associated with every business graph is used to communicate the type of After Image event for the business graph. An after image is used during data synchronization between EIS systems to capture and publish changes to business data to the WebSphere Process Server runtime. Only a non-empty verb on a business graph signals that the business graph is communicating an after image.

There are several levels of granularity to consider with an after image. If the change summary is empty, it means that the verb associated with the business graph applies to all objects in the business graph. For example, if the verb is 'Create', it means that all objects in the business object were created. Next, a business graph might contain a verb and also include a change summary with object level changes. This situation is more granular than the case with an empty change summary. Finally the most granular case is when the change summary includes both object level and property level changes.

The five standard verbs for business graphs are create, update, delete, updatewithdelete, and retrieve. The set of verbs associated with a particular type of business graph can be extended beyond the standard set.



Like the change summary and verb, the event summary is also one of the enriched capabilities provided by the business graph. To understand the event summary it is important to have a little background information on ICS business objects. In ICS, business objects included instance metadata using a verb and ObjectEventID. This information was included along with the other properties of the business object. In this release of WebSphere Process Server, this instance metadata is no longer carried in the business object, but is included as part of the enriched capabilities provided by the business graph. The verb information and change summary information carries the standard verbs for objects and object properties. The event summary is needed in order to carry non-standard verbs for each object in the business graph, along with a unique identifier (ObjectEventId).

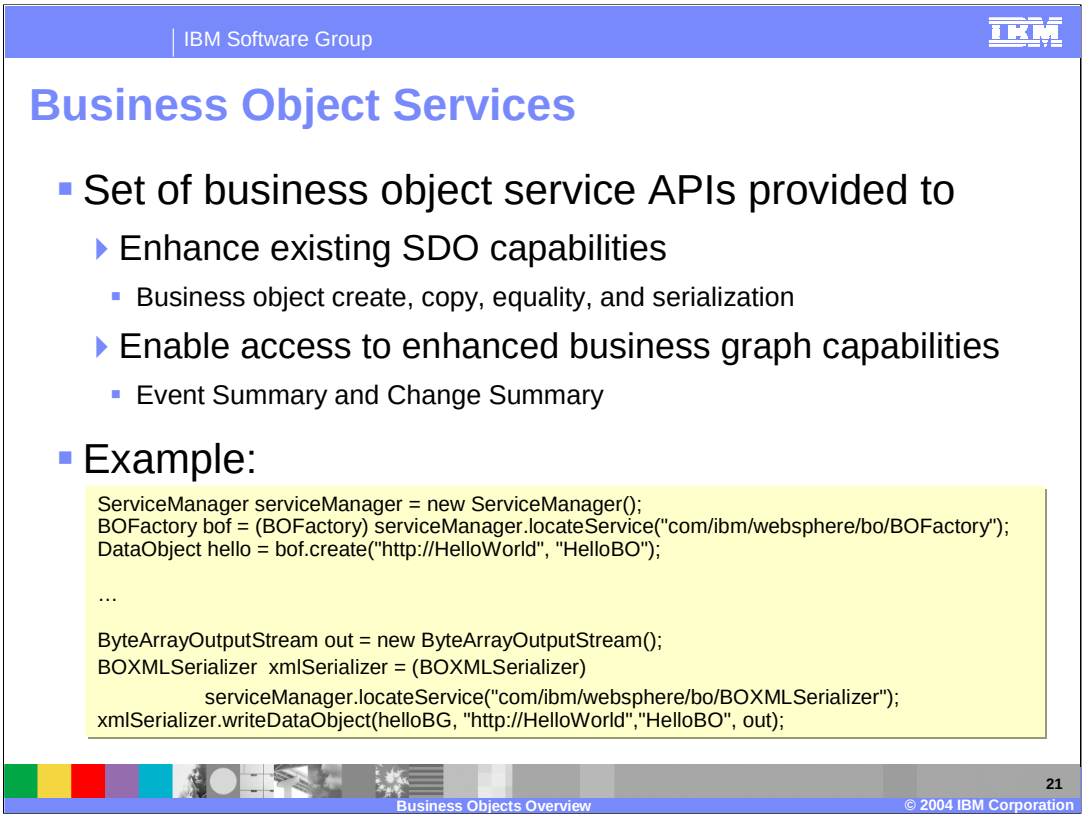


After an introduction to business objects and business graphs, you might wonder when a set of business objects should be wrapped in a business graph. To answer this question it is useful to return to the three data patterns introduced at the beginning of this presentation.

In the case of the document pattern, the solution only needs a simple XML to Java binding. This case is one that is a common pattern for many SCA applications. In this situation, the business object is used as the data abstraction within the SCA runtime. In this case, there is no need for the enhanced capabilities provided by the business graph.

For a disconnected data pattern, the change summary is needed to provide the appropriate level of data concurrency control. In this situation, a business graph is needed to provide the functionality provided by the change summary information. However, other business graph capabilities would not be used. Specifically, there would be no need for the event summary or the verb attribute.

In the final case of the event summary, the solution must be able to perform data synchronization to downstream EIS systems. For this, the full capabilities provided by the business graph are needed to communicate this information.



Several business object services are provided by the business object framework that you should be aware of. These services are a set of business object APIs provided to enhance some of the existing SDO v1.0 capabilities. In particular, there are APIs available to create, copy, show equality, and serialize and de-serialize business objects. There are also APIs that allow developers to access a business graph event or change summary. These APIs are needed because the event summary and change summary cannot be accessed like a typical business object attribute with the DataObject APIs.

This slide shows two examples of using the business object APIs. The top code snippet highlights using the BOFactory interface to create a new business object from an XML schema definition. The code example at the bottom illustrates using the BOXMLSerializer to serialize a business object to XML. In both of these examples, you use the service manager to locate the particular service that you are using.

Agenda

- Overview
- Architecture
- **Summary and References**

This section will provide a summary of business objects.

Business Object Framework: Summary

- Important part of the SCA solution in WebSphere Process Server V6
  - Abstraction layer for data access
- Built upon SDO technology
  - Enhanced to provide additional capabilities
- Provides patterns and capabilities similar to the ICS business object

The business object framework is an important part of the SCA solution in WebSphere Process Server V6.0 because it provides the abstraction layer for the data access. This framework is built upon SDO V1.0 technology, but several additional APIs are included to provide additional capabilities not found in this version of SDO. Finally, the business object framework provides support for patterns and capabilities similar to the ICS business object.

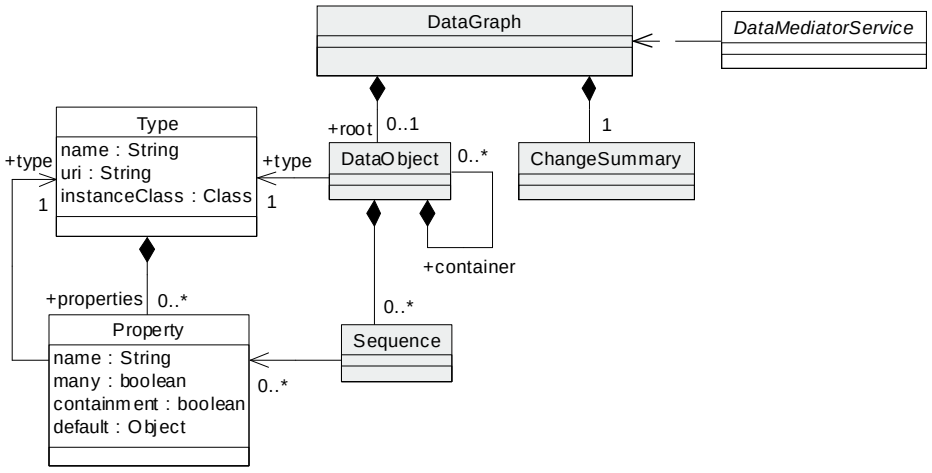
References

- XML Schema
  - http://www.w3.org/XML/Schema
- Service Data Objects (SDO)
  - http://www-128.ibm.com/developerworks/library/j-commonj-sdowmt/
  - http://www-106.ibm.com/developerworks/java/library/j-sdo/
- Eclipse Model Framework (EMF)
  - http://www.eclipse.org/emf/

Section

Appendix

Service Data Objects: Object Model



Trademarks, Copyrights, and Disclaimers

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

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

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

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

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

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

Linux is a registered trademark of Linus Torvalds.

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

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

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

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

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

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