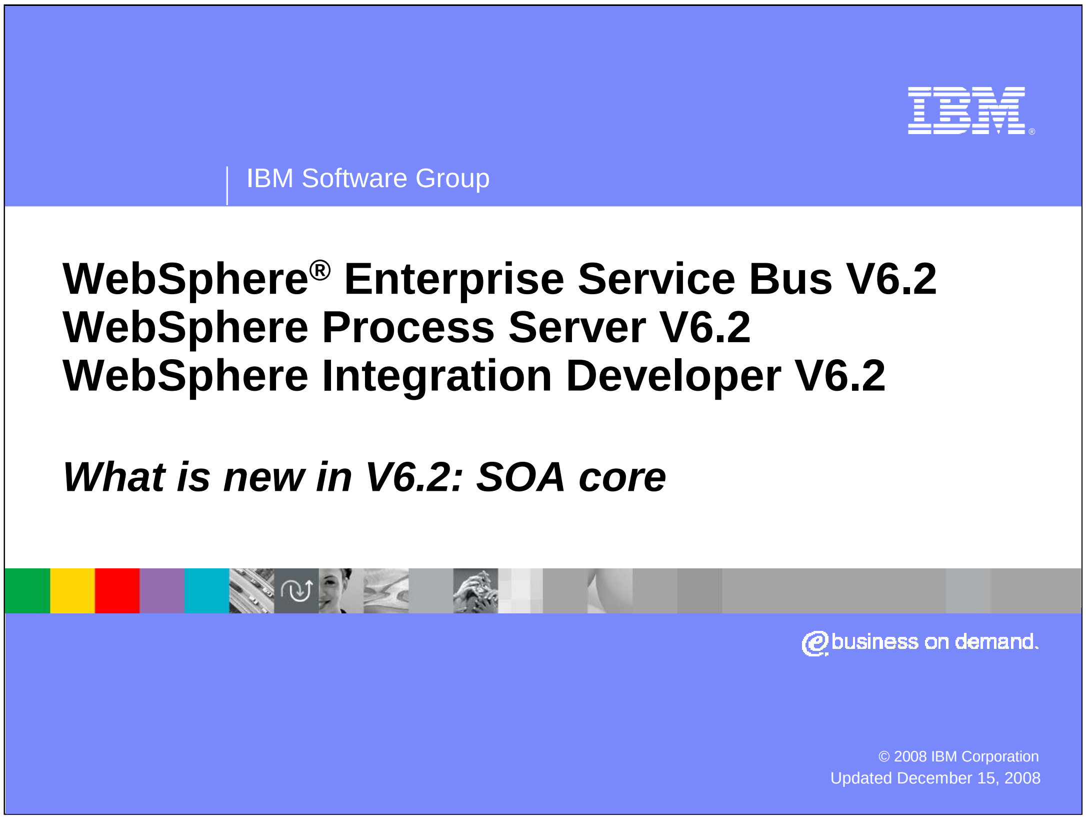


IBM Software Group

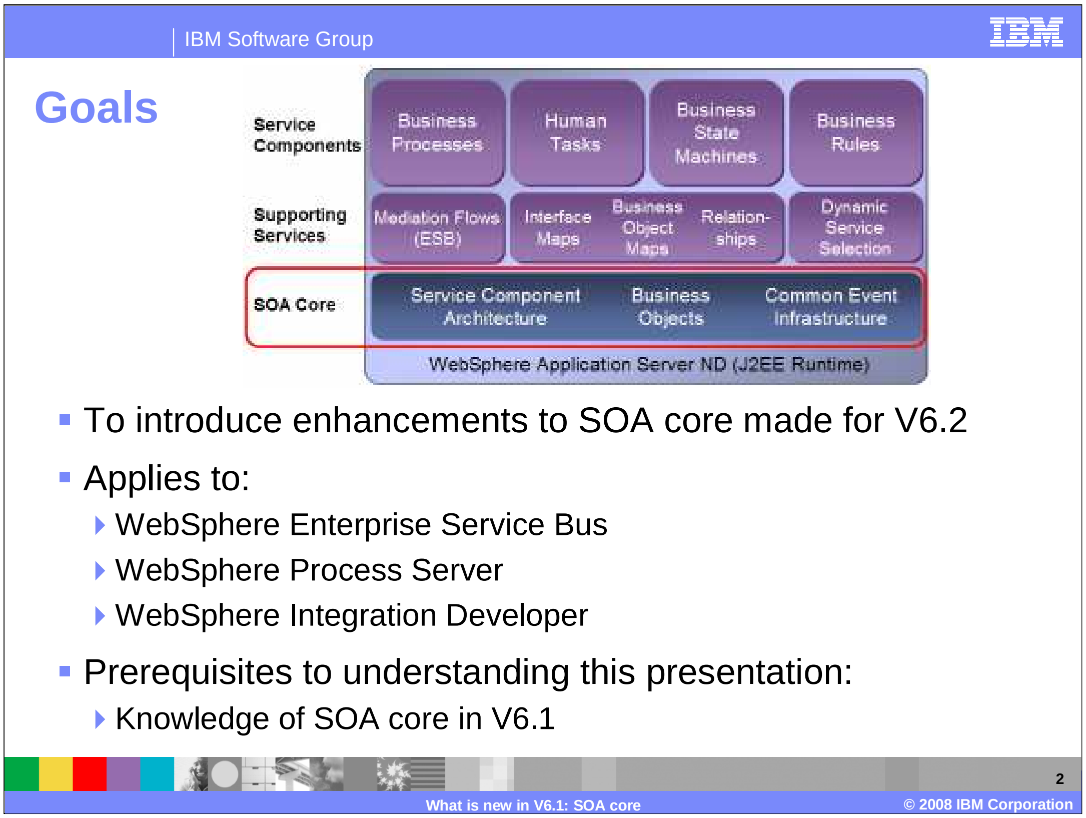
WebSphere® Enterprise Service Bus V6.2
WebSphere Process Server V6.2
WebSphere Integration Developer V6.2

What is new in V6.2: SOA core

business on demand.

Updated December 15, 2008

This presentation provides an introduction to the new function delivered in version 6.2 for the SOA core portion of WebSphere Enterprise Service Bus, WebSphere Process Server and WebSphere Integration Developer.



Goals

Service Components: Business Processes | Human Tasks | Business State Machines | Business Rules

Supporting Services: Mediation Flows (ESB) | Interface Maps | Business Object Maps | Relation-ships | Dynamic Service Selection

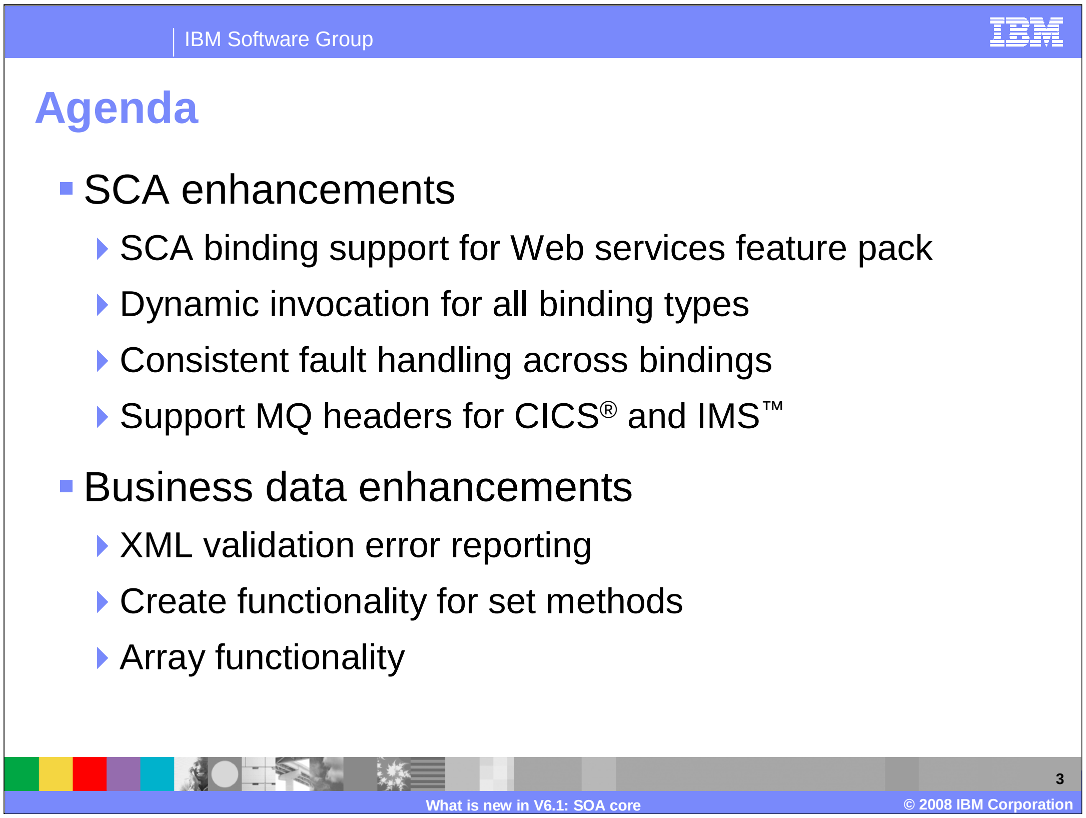
SOA Core: Service Component Architecture | Business Objects | Common Event Infrastructure

WebSphere Application Server ND (J2EE Runtime)

- To introduce enhancements to SOA core made for V6.2
- Applies to:
  - WebSphere Enterprise Service Bus
  - WebSphere Process Server
  - WebSphere Integration Developer
- Prerequisites to understanding this presentation:
  - Knowledge of SOA core in V6.1

The goal of this presentation is to introduce you to the enhancements that have been made to the service oriented architecture core for version 6.2. As you can see in the architecture chart, the SOA core is composed of the service component architecture, business objects and the common event infrastructure. These enhancements apply equally to WebSphere Process Server and WebSphere Enterprise Service Bus, including new and changed capabilities in WebSphere Integration Developer which support these enhancements from a development tool perspective.

In order to understand the material in this presentation you should already have a knowledge of the capabilities of SOA core in version 6.1 of the products.



IBM Software Group

IBM

Agenda

- SCA enhancements
  - SCA binding support for Web services feature pack
  - Dynamic invocation for all binding types
  - Consistent fault handling across bindings
  - Support MQ headers for CICS® and IMS™
- Business data enhancements
  - XML validation error reporting
  - Create functionality for set methods
  - Array functionality

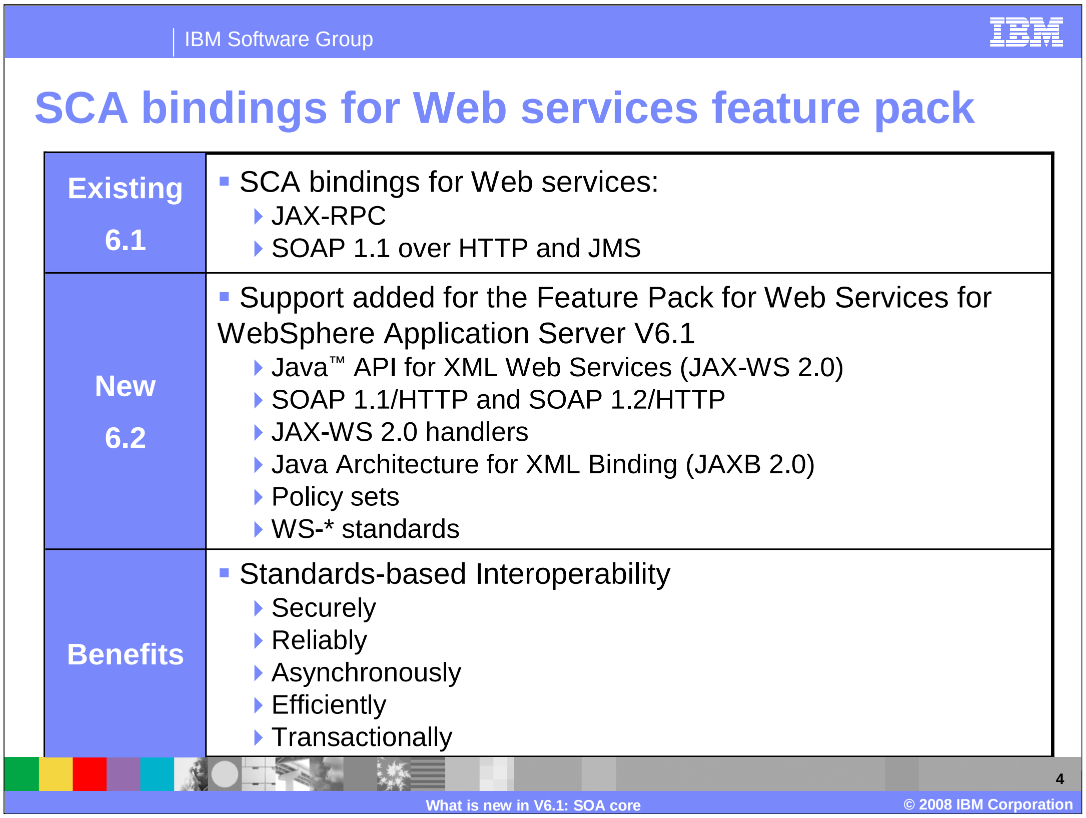
3

What is new in V6.1: SOA core

© 2008 IBM Corporation

The presentation first looks at the enhancements for service component architecture. The most significant of these is the addition of SCA bindings that support the Web services feature pack, which implements support for JAX-WS Web services. The next enhancement is the support for dynamic invocation across all binding types, including messaging and HTTP bindings. Fault handling has been enhanced with a consistent approach that works across the various types of bindings, with support for both business and server runtime exceptions. The last of the SCA enhancements is the ability for an import to be configured to statically set values into MQ headers for interaction with CICS and IMS.

The next set of enhancements relate to the handling of business objects. A consistent mechanism has been implemented for reporting XML validation errors with error logging that specifically identifies the error encountered. The DataObject APIs have been enhanced to improve the usability of the set methods when used with complex nested business objects. Finally, an approach to providing array like functionality for XML has been adopted.



IBM Software Group

SCA bindings for Web services feature pack

Existing 6.1	▪ SCA bindings for Web services: ▸ JAX-RPC ▸ SOAP 1.1 over HTTP and JMS
New 6.2	▪ Support added for the Feature Pack for Web Services for WebSphere Application Server V6.1 ▸ Java™ API for XML Web Services (JAX-WS 2.0) ▸ SOAP 1.1/HTTP and SOAP 1.2/HTTP ▸ JAX-WS 2.0 handlers ▸ Java Architecture for XML Binding (JAXB 2.0) ▸ Policy sets ▸ WS-* standards
Benefits	▪ Standards-based Interoperability ▸ Securely ▸ Reliably ▸ Asynchronously ▸ Efficiently ▸ Transactionally

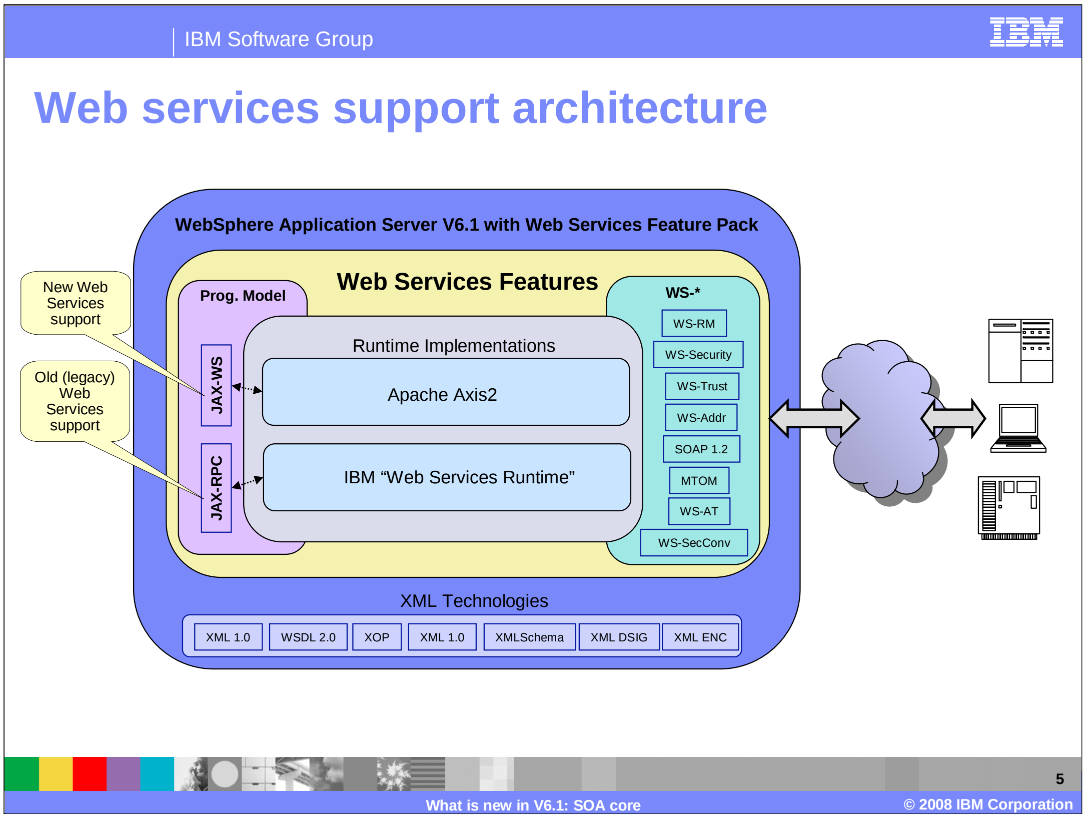
4

What is new in V6.1: SOA core

© 2008 IBM Corporation

This enhancement provides support for JAX-WS Web services. In version 6.1, the Web services support was for JAX-RPC with SOAP 1.1 over HTTP or JMS. In version 6.2 support has been added to enable SCA bindings which use JAX-WS as provided by the Feature Pack for Web Services for WebSphere Application Server 6.1. This is a significant in that Web service support is now consistent with the latest Web service standards adopted by the industry. This includes the Java API for XML Web Services (JAX-WS 2.0), SOAP 1.1 over HTTP and SOAP 1.2 over HTTP. Additionally, JAX-WS 2.0 handlers are supported, as is Java Architecture for XML Binding (JAXB 2.0), the use of policy sets and support for several of the Web service standards.

Among the benefits of this enhancement is enhanced security, reliability and transactionality for Web service interactions. Additionally, there is better interoperability with other platforms implementing Web services.



IBM Software Group

Web services support architecture

WebSphere Application Server V6.1 with Web Services Feature Pack

Web Services Features

Prog. Model

JAX-WS

JAX-RPC

Runtime Implementations

Apache Axis2

IBM “Web Services Runtime”

WS-*

WS-RM

WS-Security

WS-Trust

WS-Addr

SOAP 1.2

MTOM

WS-AT

WS-SecConv

XML Technologies

XML 1.0	WSDL 2.0	XOP	XML 1.0	XMLSchema	XML DSIG	XML ENC

New Web Services support

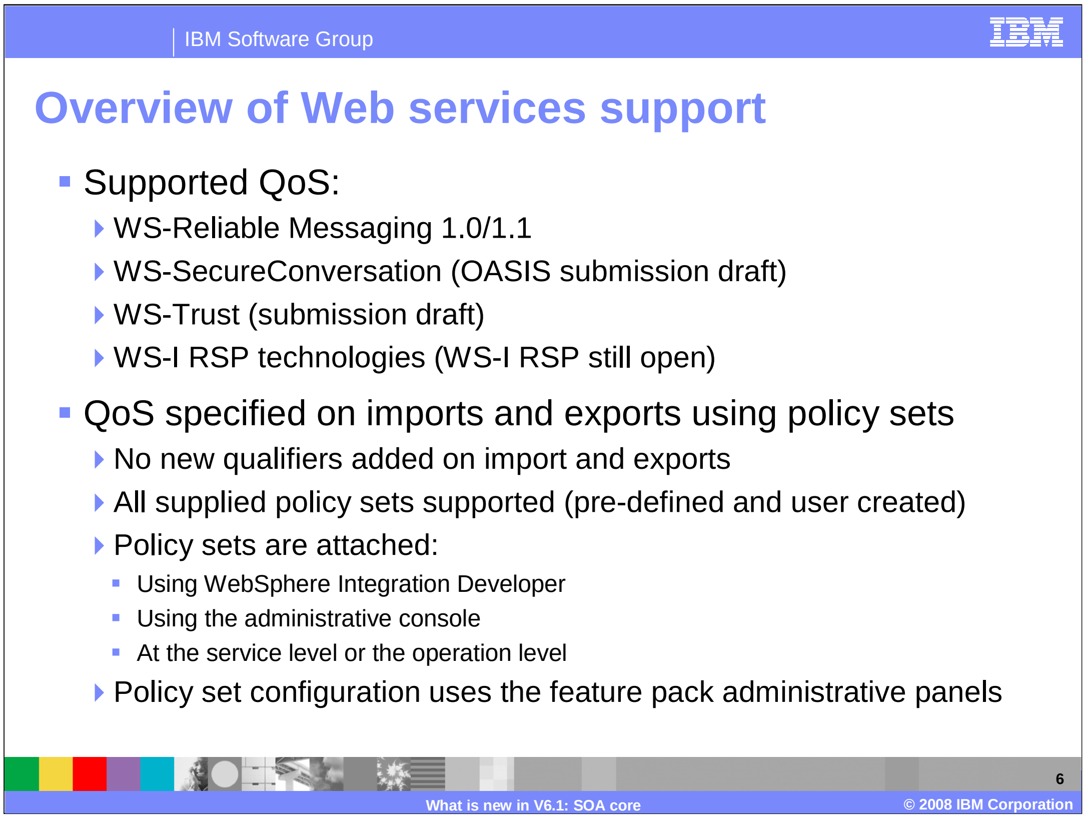
Old (legacy) Web Services support

5

What is new in V6.1: SOA core

© 2008 IBM Corporation

This slide provides an architectural view of Web services support in WebSphere Application Server when the Feature Pack for Web Services for WebSphere Application Server V6.1 is installed. You can see that the older legacy style is JAX-RPC and makes use of the IBM Web services runtime engine, whereas the new style JAX-WS makes use of the Apache Axis2 engine. When used with WebSphere Process Server and WebSphere Enterprise Service Bus version 6.2, not all of the capabilities shown here are used by the support for Web services in SCA. The following slides provide an overview of what parts of this architecture are directly used by SCA Web service bindings.



IBM Software Group

Overview of Web services support

- Supported QoS:
  - WS-Reliable Messaging 1.0/1.1
  - WS-SecureConversation (OASIS submission draft)
  - WS-Trust (submission draft)
  - WS-I RSP technologies (WS-I RSP still open)
- QoS specified on imports and exports using policy sets
  - No new qualifiers added on import and exports
  - All supplied policy sets supported (pre-defined and user created)
  - Policy sets are attached:
    - Using WebSphere Integration Developer
    - Using the administrative console
    - At the service level or the operation level
  - Policy set configuration uses the feature pack administrative panels

6

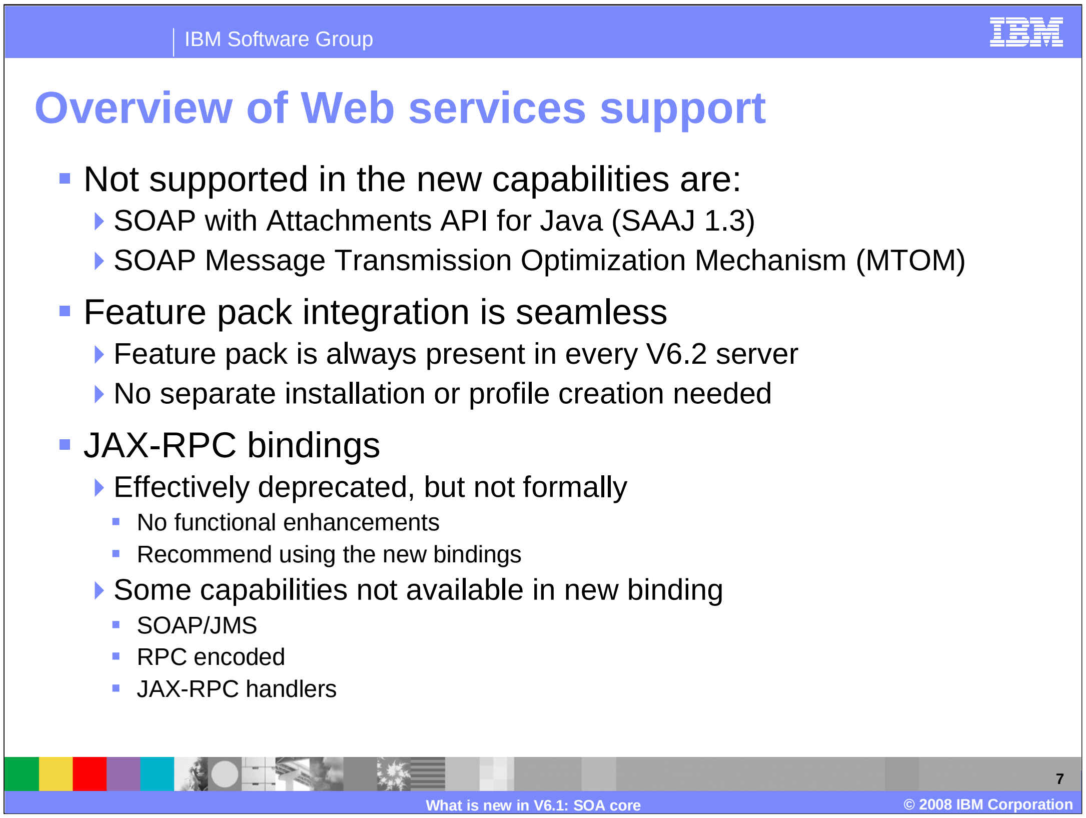
What is new in V6.1: SOA core

© 2008 IBM Corporation

This slide and the next look at the Web service support in WebSphere Process Server and WebSphere Enterprise Service Bus version 6.2 using the SCA Web service bindings. The bindings support the qualities of service defined by the WS-Reliable Messaging 1.0 and 1.1 specifications, WS-SecureConversation, WS-Trust and WS-I Reliable Secure Profile. The Web service standards are continuing to evolve and therefore, as noted in the slide, support might be based on a standard that is still being defined.

No new qualifiers have been added to the imports or exports to support these qualities of service. They are supported by SCA bindings through the use of policy sets. Any pre-defined or user created policy set can be used. The policy sets can be attached during development using WebSphere Integration Developer and are associated with all operations for the service. They can also be attached administratively using the administrative console, and when done this way can apply to specific operations if required.

The configuration of policy sets is done using the panels provided as part of the Web service feature pack.



Overview of Web services support

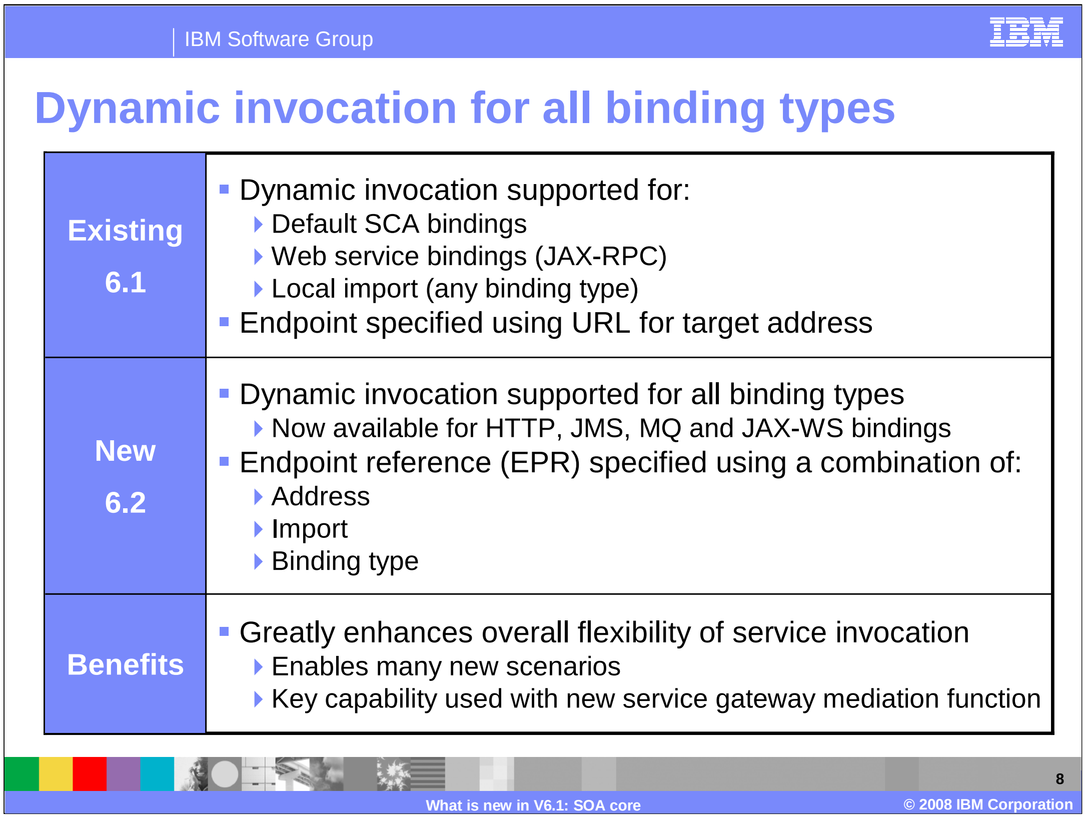
- Not supported in the new capabilities are:
  - SOAP with Attachments API for Java (SAAJ 1.3)
  - SOAP Message Transmission Optimization Mechanism (MTOM)
- Feature pack integration is seamless
  - Feature pack is always present in every V6.2 server
  - No separate installation or profile creation needed
- JAX-RPC bindings
  - Effectively deprecated, but not formally
    - No functional enhancements
    - Recommend using the new bindings
  - Some capabilities not available in new binding
    - SOAP/JMS
    - RPC encoded
    - JAX-RPC handlers

Not all of the Web service standards are supported by the Web service SCA bindings. The SOAP with Attachments API for Java is not supported and neither is the related SOAP Message Transmission Optimization Mechanism.

The installation of WebSphere Process Server and WebSphere Enterprise Service Bus version 6.2 automatically installs the Web service feature pack. This provides a seamless integration of this support and eliminates a requirement for separate installation and profile creation

For the version 6.2 release, the legacy JAX-RPC bindings are fully supported. However, the intention is to provide no further enhancements to these bindings and additionally they might be formally deprecated in a subsequent release. Therefore, it is better for you to use the new JAX-WS bindings whenever possible.

There are some capabilities of the JAX-RPC bindings which are not available with the new JAX-WS bindings. SOAP over JMS is not supported because the use of JAX-WS bindings with reliable messaging replaces the motivation behind the use of SOAP over JMS. Also, for obvious reasons, the support for RPC encoding and JAX-RPC handlers is not included in the new bindings.



IBM Software Group

Dynamic invocation for all binding types

Existing **6.1**	▪ Dynamic invocation supported for: ▸ Default SCA bindings ▸ Web service bindings (JAX-RPC) ▸ Local import (any binding type) ▪ Endpoint specified using URL for target address
New **6.2**	▪ Dynamic invocation supported for all binding types ▸ Now available for HTTP, JMS, MQ and JAX-WS bindings ▪ Endpoint reference (EPR) specified using a combination of: ▸ Address ▸ Import ▸ Binding type
Benefits	▪ Greatly enhances overall flexibility of service invocation ▸ Enables many new scenarios ▸ Key capability used with new service gateway mediation function

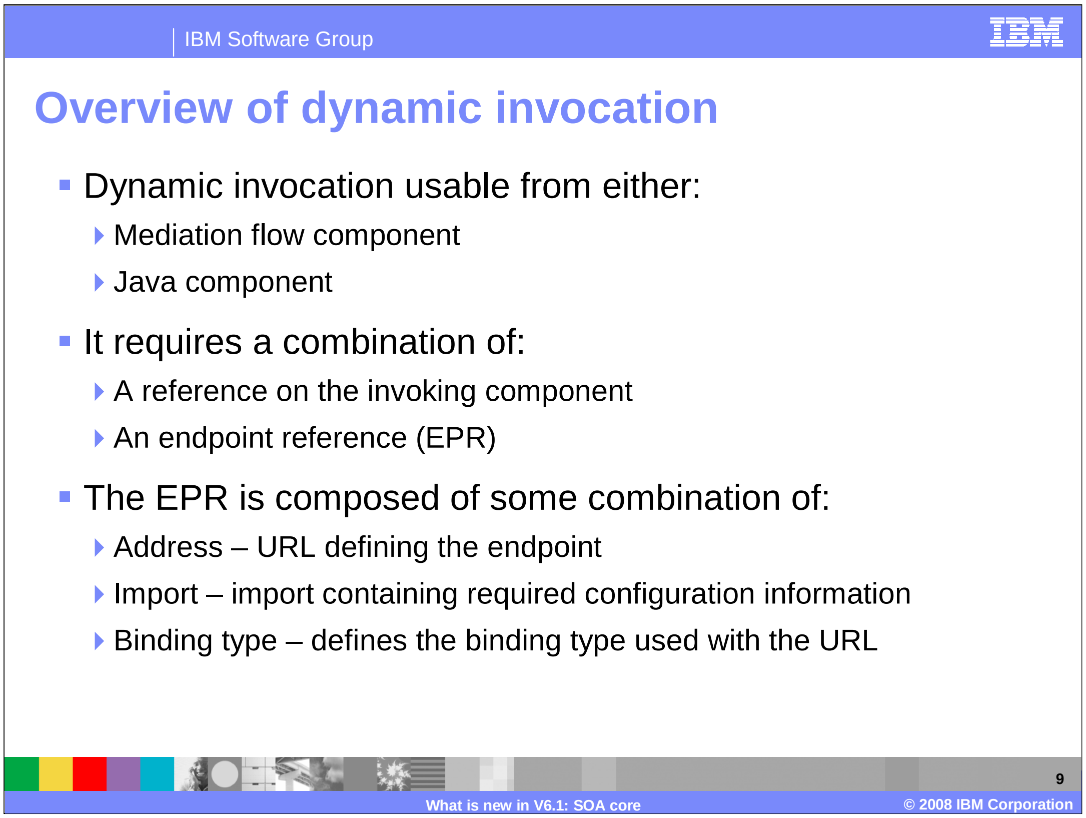
8

What is new in V6.1: SOA core

© 2008 IBM Corporation

This enhancement provides dynamic invocation capability that can be used with SCA bindings of all types. In version 6.1 the dynamic invocation support was limited to the default SCA bindings, JAX-RPC Web service bindings and reference to a local import which might be of any binding type. In all these cases, the endpoint was specified using a URL for the target address. With the introduction of the new support in version 6.2, dynamic invocation can be used with all of the SCA binding types, such as messaging, HTTP and JAX-WS Web service bindings. Because a URL is not sufficient to specify all that is needed to connect with some endpoints, there needs to be the capability to combine a configured import with a URL to fully describe some endpoints. To do this, the endpoint is specified as an endpoint reference object that contains a URL for the address, the name of an import and a designation of the type of binding. Not all of these need to be specified. Different combinations of these along with a reference on the calling component combine to provide varying behaviors for different circumstances.

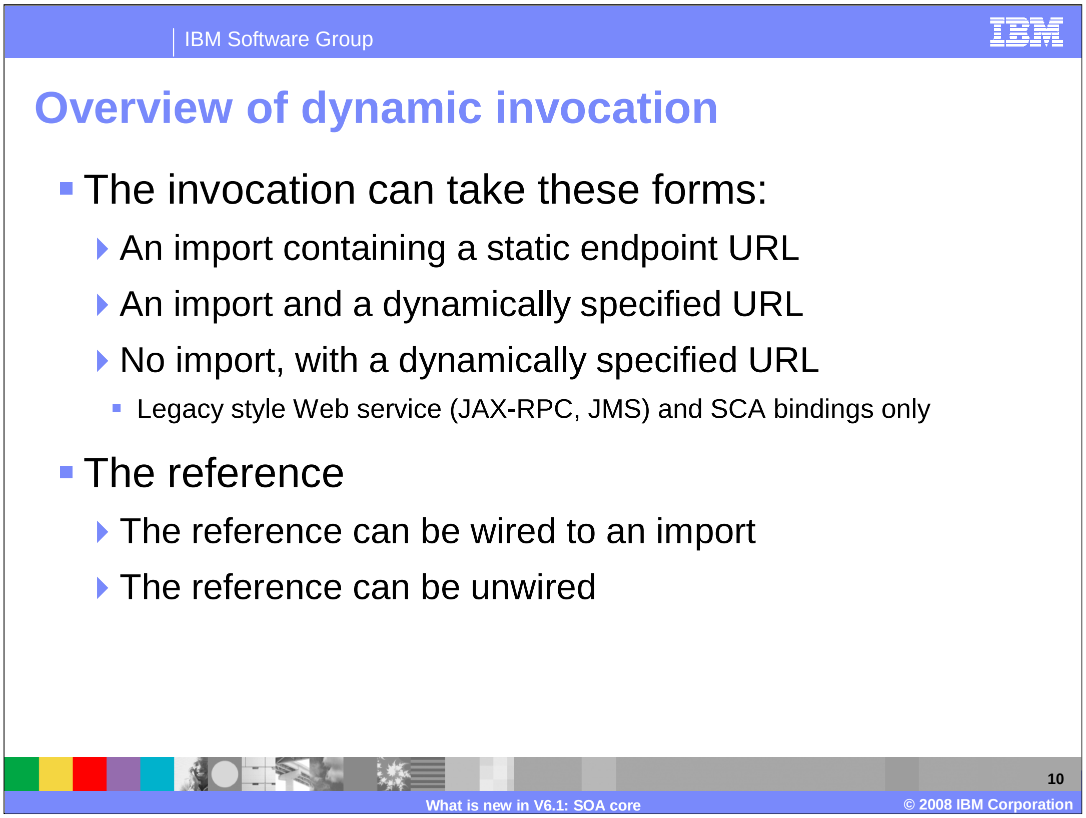
This functionality greatly enhances the overall flexibility of service invocation and enables many new scenarios. One of the key capabilities for which dynamic invocation is a critical part is the new service gateway function provided for mediation flows.



Overview of dynamic invocation

- Dynamic invocation usable from either:
  - Mediation flow component
  - Java component
- It requires a combination of:
  - A reference on the invoking component
  - An endpoint reference (EPR)
- The EPR is composed of some combination of:
  - Address – URL defining the endpoint
  - Import – import containing required configuration information
  - Binding type – defines the binding type used with the URL

Dynamic invocation can be used from within a Java component using an API and can also be done within a mediation flow component with either a service invoke primitive or a callout. Dynamic invocation requires the specification of a reference that exists on the invoking Java component or mediation flow component. Additionally, an endpoint reference must be specified which contains some combination of an address, name of an import and designation of the binding type.



IBM Software Group

Overview of dynamic invocation

- The invocation can take these forms:
  - An import containing a static endpoint URL
  - An import and a dynamically specified URL
  - No import, with a dynamically specified URL
    - Legacy style Web service (JAX-RPC, JMS) and SCA bindings only
- The reference
  - The reference can be wired to an import
  - The reference can be unwired

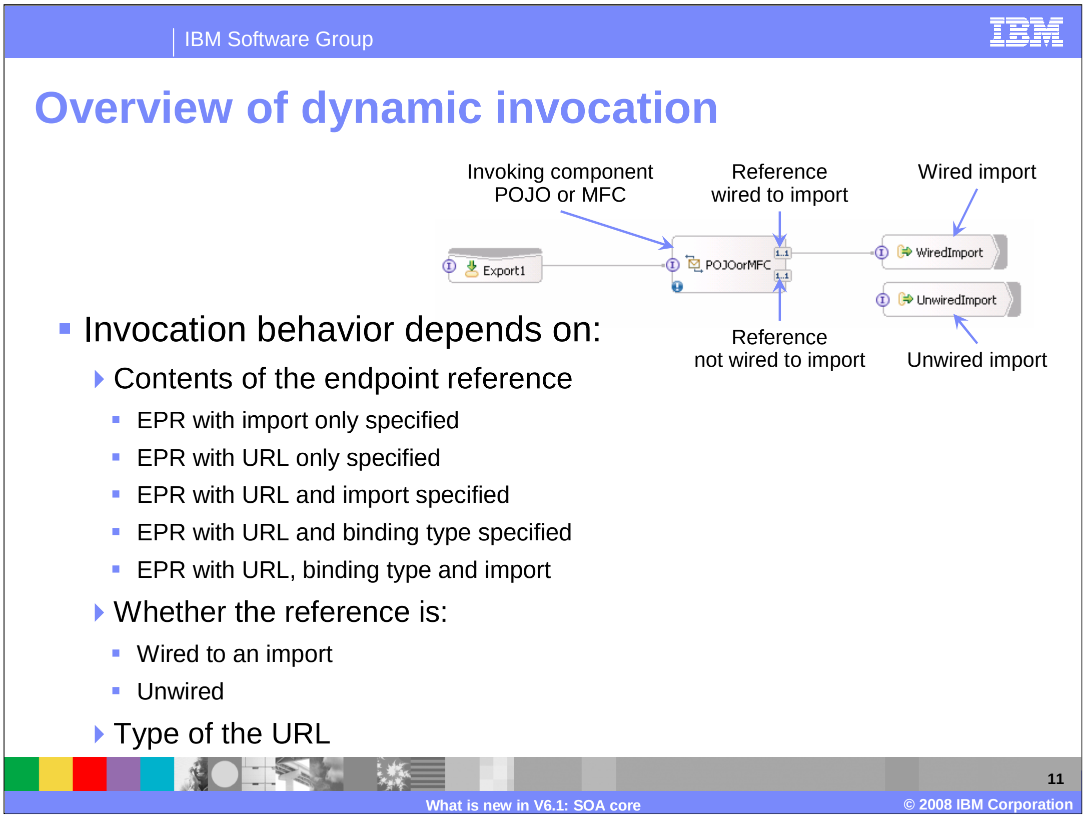
10

What is new in V6.1: SOA core

© 2008 IBM Corporation

There are three basic forms that the dynamic invocation can have. There can be an import which statically contains the endpoint URL, there can be an import used with a URL that is dynamically specified, and there can be a URL without an import. This last case results in the equivalent support that was provided before version 6.2, supporting legacy style Web services and default SCA bindings.

The reference on the invoking component does not have to be wired to an import. When it is, that import is used unless an import is specified in the endpoint reference.


IBM Software Group

Overview of dynamic invocation

Invoking component POJO or MFC

Reference wired to import

Wired import

Reference not wired to import

Unwired import

- Invocation behavior depends on:
  - Contents of the endpoint reference
    - EPR with import only specified
    - EPR with URL only specified
    - EPR with URL and import specified
    - EPR with URL and binding type specified
    - EPR with URL, binding type and import
  - Whether the reference is:
    - Wired to an import
    - Unwired
  - Type of the URL

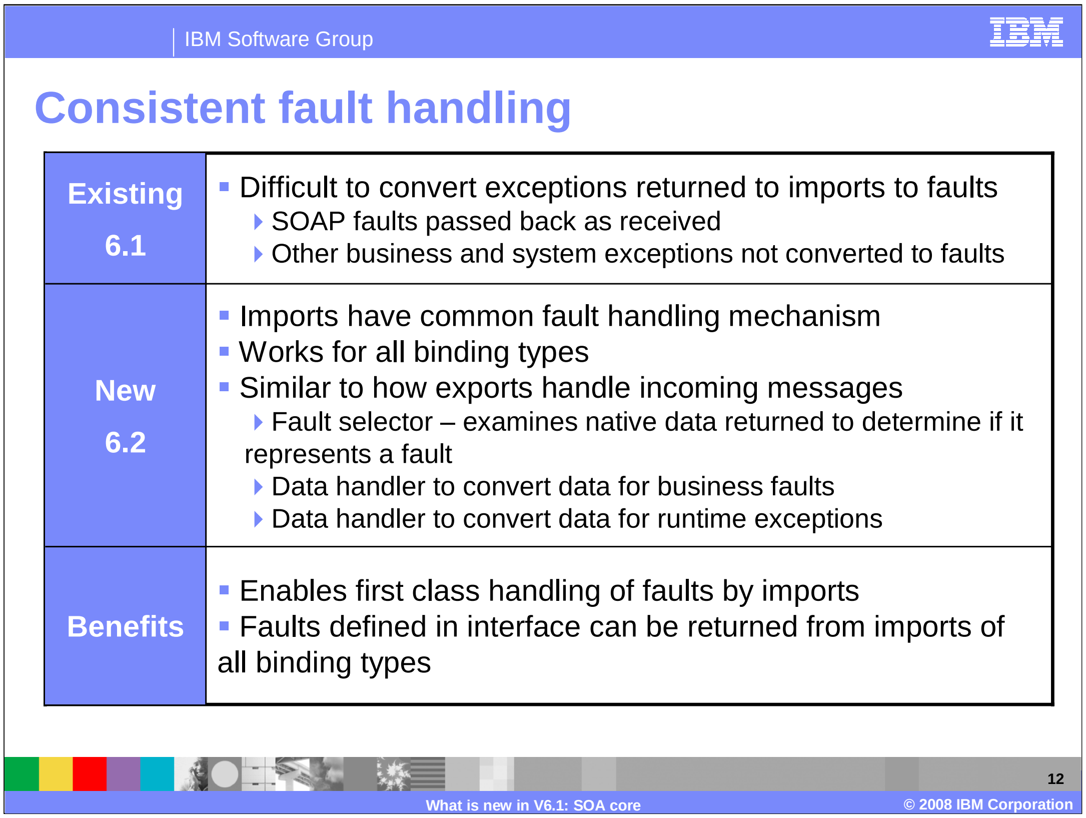
11

What is new in V6.1: SOA core

© 2008 IBM Corporation

This slide shows an SCA assembly diagram that illustrates that when dynamic invocation is used, you can have references on the invoking component that are wired or unwired. Also, imports can be wired to a reference or remain unwired.

The behavior of the invocation varies depending upon the contents of the endpoint reference which can have any combination of the three attributes. The behavior is also dependent upon whether the reference used is wired to an import and the type of the URL. The specific behaviors for each of these variations is beyond the scope of this presentation. You can reference the Information Center for a description of the various behaviors.

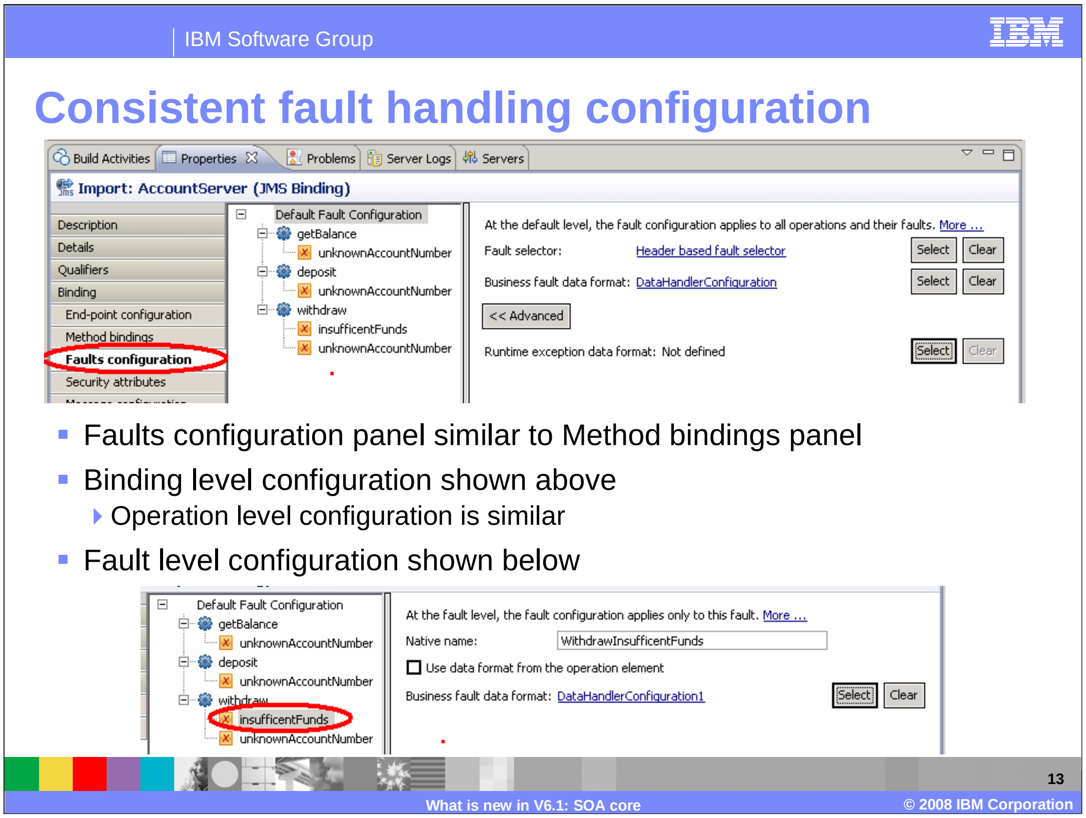


Consistent fault handling

Existing 6.1	▪ Difficult to convert exceptions returned to imports to faults ▸ SOAP faults passed back as received ▸ Other business and system exceptions not converted to faults
New 6.2	▪ Imports have common fault handling mechanism ▪ Works for all binding types ▪ Similar to how exports handle incoming messages ▸ Fault selector – examines native data returned to determine if it represents a fault ▸ Data handler to convert data for business faults ▸ Data handler to convert data for runtime exceptions
Benefits	▪ Enables first class handling of faults by imports ▪ Faults defined in interface can be returned from imports of all binding types

This enhancement provides a consistent approach to handling faults across the various kinds of SCA bindings. In version 6.1 it is difficult to convert exceptions that are returned to imports into faults. For Web service bindings, a SOAP fault received by an import is passed back as received. However, other business and system exceptions are not converted to faults, and the resulting behavior varies by binding type. With the introduction of consistent fault handling in version 6.2, imports now have a common mechanism for handling business and system exceptions that is the same across all of the binding types. The support is similar to how exports handle incoming messages. There is a fault selector which examines the native data being returned to the import which then decides if the data represents a fault. If so, there is a data handler configured which can convert the native data into a fault. There are separate data handlers for business faults and for runtime exceptions because the transformation of the native data might need to be different for those cases.

Among the benefits of this enhancement is that imports now provide first class support for the handling of faults, allowing faults defined on the interface to be returned by imports of all binding types.



Consistent fault handling configuration

Import: AccountServer (JMS Binding)

Default Fault Configuration
- getBalance
  - unknownAccountNumber
- deposit
  - unknownAccountNumber
- withdraw
  - insufficentFunds
  - unknownAccountNumber

At the default level, the fault configuration applies to all operations and their faults. More ...

Fault selector: Header based fault selector

Business fault data format: DataHandlerConfiguration

Runtime exception data format: Not defined

- Faults configuration panel similar to Method bindings panel
- Binding level configuration shown above
  - Operation level configuration is similar
- Fault level configuration shown below

Default Fault Configuration
- getBalance
  - unknownAccountNumber
- deposit
  - unknownAccountNumber
- withdraw
  - insufficentFunds
  - unknownAccountNumber

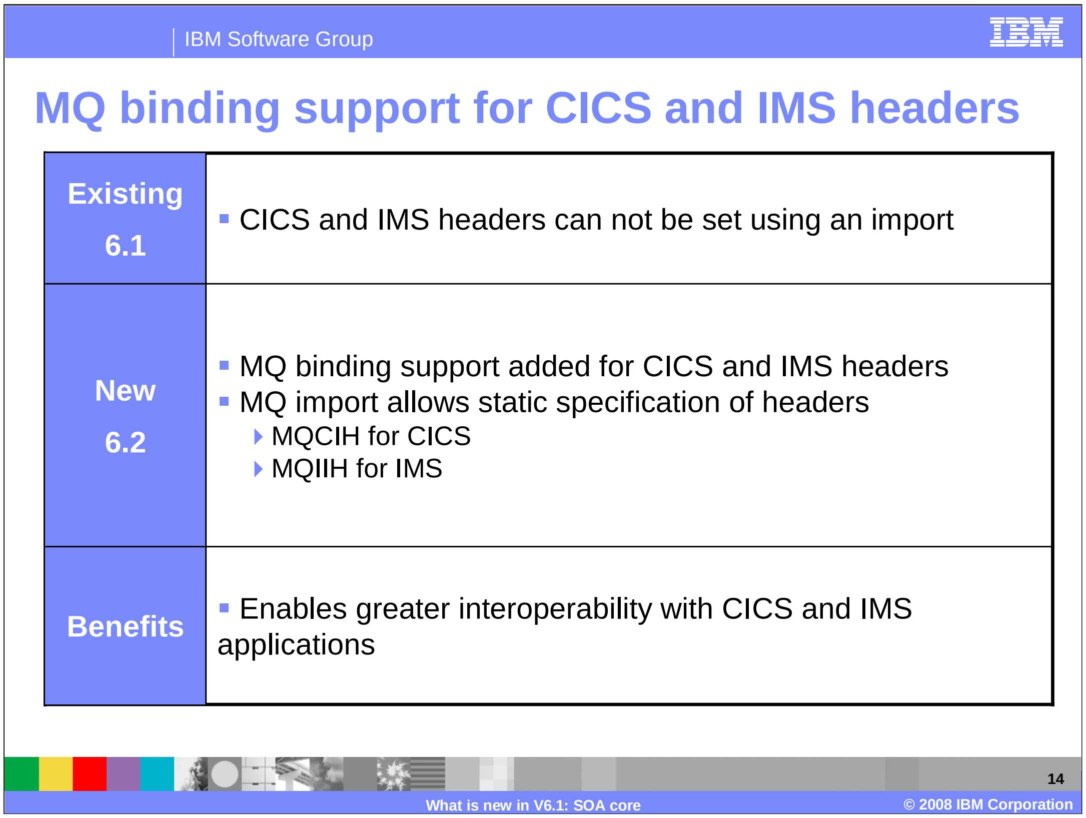
At the fault level, the fault configuration applies only to this fault. More ...

Native name: WithdrawInsufficentFunds

☐ Use data format from the operation element

Business fault data format: DataHandlerConfiguration1

This slide shows a screen capture of the faults configuration panel in the properties view of an import. The top panel is where the fault handling is configured for all of the operations associated with this import binding. There is a similar panel provided for each of the operations which allow operation specific configuration of the fault selector and data handling for business and runtime faults. There is also the ability to configure at the individual fault level, as shown at the bottom of the slide. The native name specifies the value returned by the fault selector that indicates this is the fault being raised. There is the ability at this level to configure data handling for business exceptions but not for runtime exceptions.



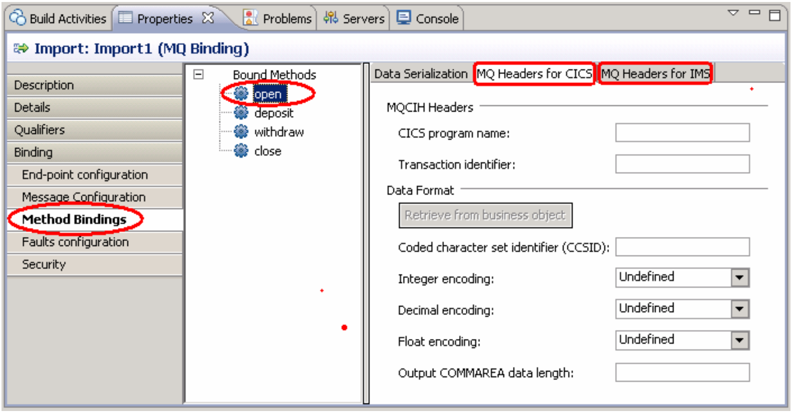
MQ binding support for CICS and IMS headers

Existing 6.1	▪ CICS and IMS headers can not be set using an import
New 6.2	▪ MQ binding support added for CICS and IMS headers ▪ MQ import allows static specification of headers ▸ MQCIH for CICS ▸ MQIIH for IMS
Benefits	▪ Enables greater interoperability with CICS and IMS applications

This enhancement provides the ability to statically initialize MQ headers for CICS and IMS from an import with an MQ binding. In version 6.1 this capability was not provided by an import binding. With this enhancement in version 6.2, it is easy to specify values for the MQCIH header used by CICS or the MQIIH header used by IMS, thus enabling greater interoperability with CICS and IMS applications.

MQ binding support for CICS and IMS headers

- Applied at the operation level in method bindings



Import: Import1 (MQ Binding)

MQCIH Headers

CICS program name:

Transaction identifier:

Data Format

Coded character set identifier (CCSID):

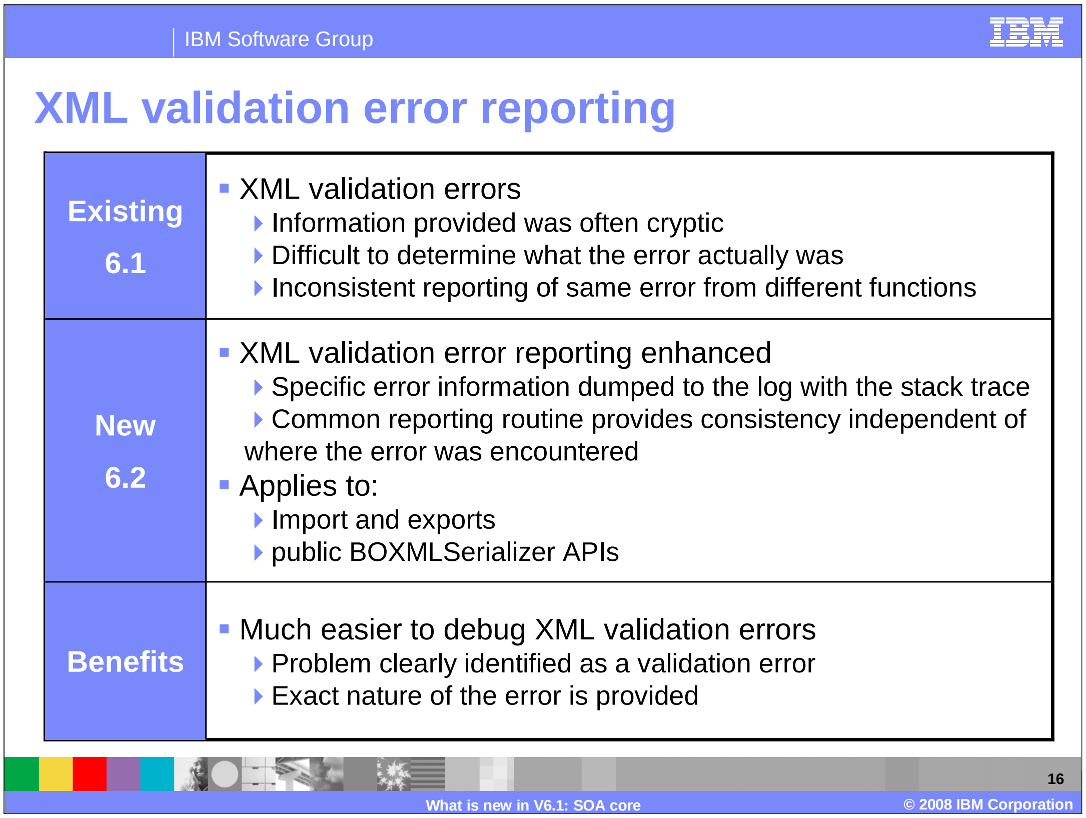
Integer encoding: Undefined

Decimal encoding: Undefined

Float encoding: Undefined

Output COMMAREA data length:

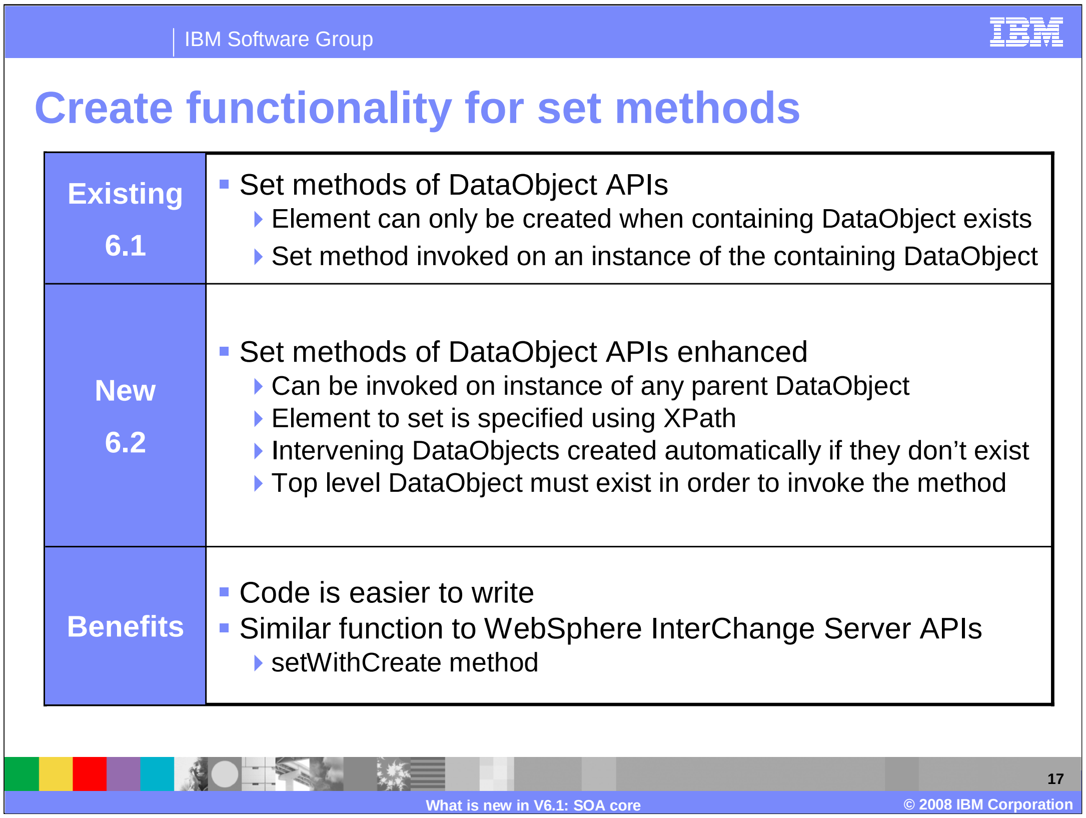
This is a screen capture from the properties view of an import binding. For each individual operation, there is a separate panel for the MQCIH header and the MQIIH header. These panels are part of the method bindings and are associated with specific operations on the interface. Setting these values at the binding level is not provided because the settings are typically operation specific.



XML validation error reporting

Existing **6.1**	▪ XML validation errors ▸ Information provided was often cryptic ▸ Difficult to determine what the error actually was ▸ Inconsistent reporting of same error from different functions
New **6.2**	▪ XML validation error reporting enhanced ▸ Specific error information dumped to the log with the stack trace ▸ Common reporting routine provides consistency independent of where the error was encountered ▪ Applies to: ▸ Import and exports ▸ public BOXMLSerializer APIs
Benefits	▪ Much easier to debug XML validation errors ▸ Problem clearly identified as a validation error ▸ Exact nature of the error is provided

This enhancement provides consistency in the reporting of XML validation errors. In version 6.1 the information provided when an XML validation error occurred was often cryptic and might be difficult to interpret to determine what the actual error was. In addition, the same error might have different results depending upon which function discovered the problem. With the introduction of consistent error reporting in version 6.2 there is a common error reporting routine that gets called when an XML validation error is encountered. This results in specific error information being dumped to the system log along with a stack trace identifying where the error was encountered. This applies to errors found in imports and exports of varying binding types and to all uses of the public BOXMLSerializer APIs. The benefit of this enhancement is that XML validation errors are much easier to debug because the information provided is consistent and identifies what is wrong in the XML that caused the validation to fail.



IBM Software Group

IBM

Create functionality for set methods

Existing 6.1	▪ Set methods of DataObject APIs ▸ Element can only be created when containing DataObject exists ▸ Set method invoked on an instance of the containing DataObject
New 6.2	▪ Set methods of DataObject APIs enhanced ▸ Can be invoked on instance of any parent DataObject ▸ Element to set is specified using XPath ▸ Intervening DataObjects created automatically if they don't exist ▸ Top level DataObject must exist in order to invoke the method
Benefits	▪ Code is easier to write ▪ Similar function to WebSphere InterChange Server APIs ▸ setWithCreate method

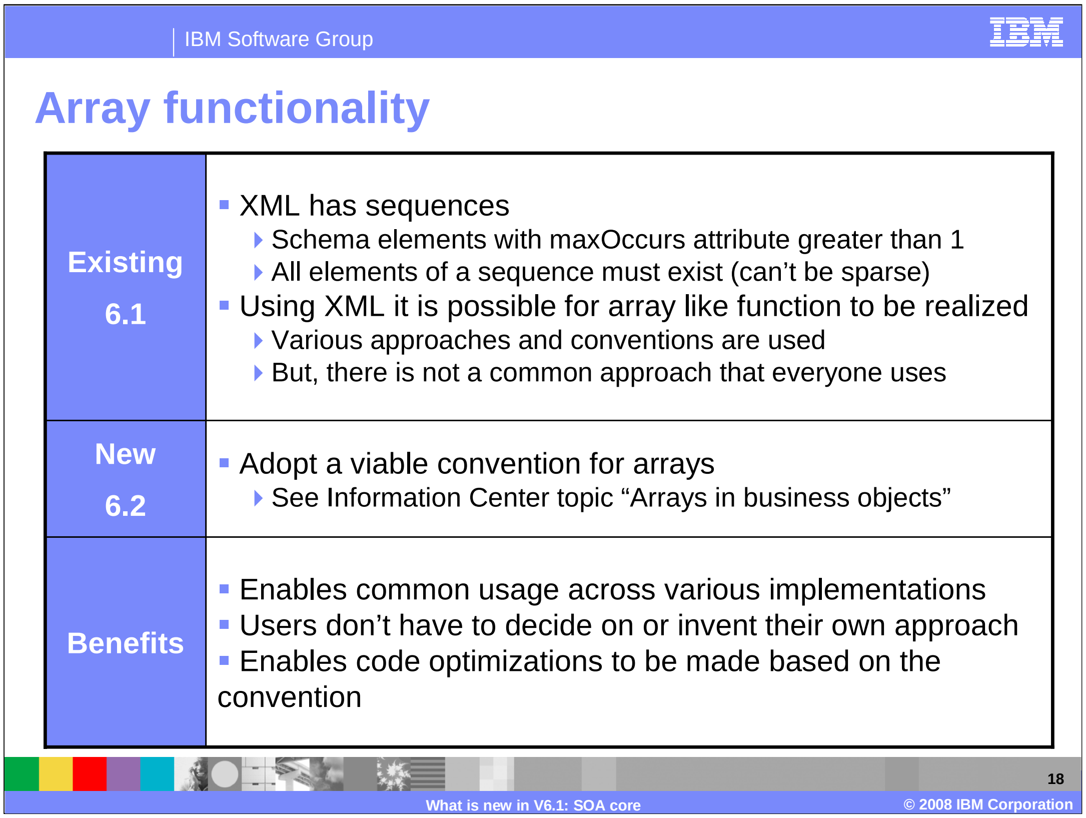
17

What is new in V6.1: SOA core

© 2008 IBM Corporation

This enhancement provides increased usability for the set methods provided by the DataObject APIs when dealing with complex data types with nested data objects. In version 6.1 when setting the value of an element within a data object, the method had to be invoked on the data object containing the element. Therefore, that data object needed to exist before setting the contained element. With the introduction of create functionality for set methods in version 6.2 a set method can be invoked on an instance of any parent data object. The element to set is identified using an XPath expression. If there are intervening data objects which do not yet exist, they are automatically created. At a minimum, the top level data object must exist as there must be an existing data object on which to invoke the set method.

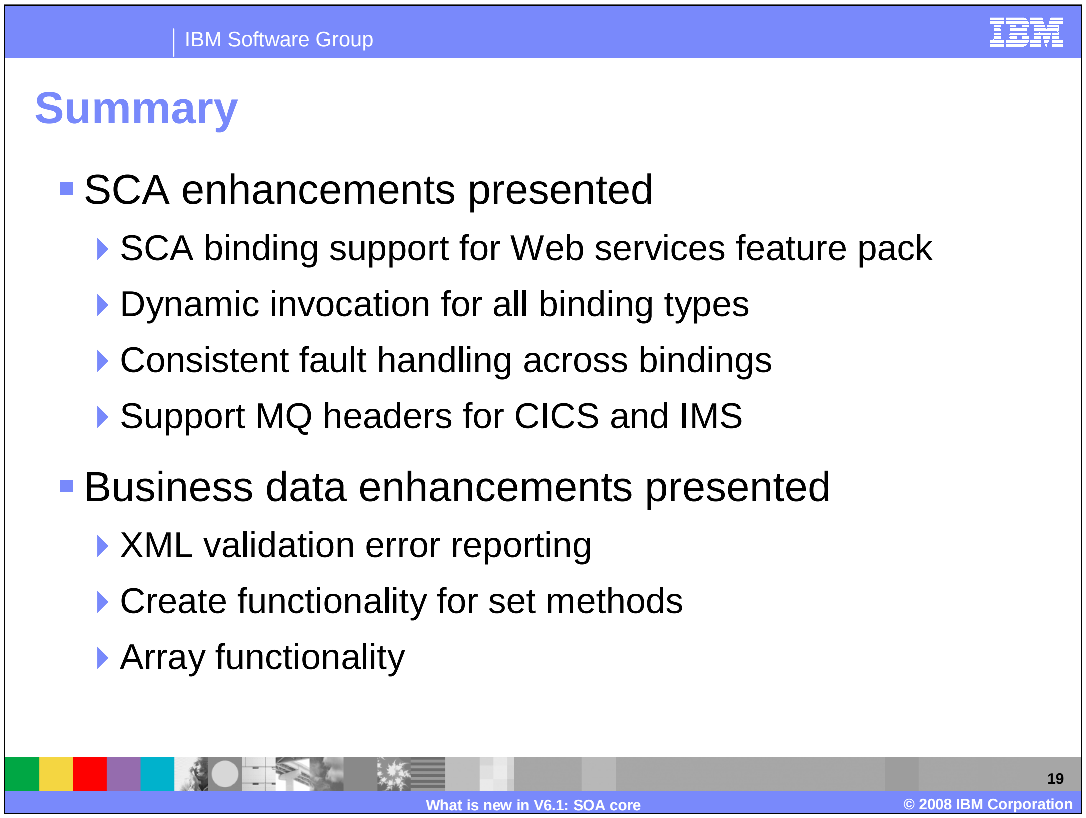
Among the benefits of this enhancement is that code working with complex data object structures in much easier to write. This is a similar functionality to the setWithCreate methods which are APIs provided by the WebSphere InterChange Server, which is one of the predecessor products to WebSphere Process Server.



Array functionality

Existing 6.1	▪ XML has sequences ▸ Schema elements with maxOccurs attribute greater than 1 ▸ All elements of a sequence must exist (can't be sparse) ▪ Using XML it is possible for array like function to be realized ▸ Various approaches and conventions are used ▸ But, there is not a common approach that everyone uses
New 6.2	▪ Adopt a viable convention for arrays ▸ See Information Center topic "Arrays in business objects"
Benefits	▪ Enables common usage across various implementations ▪ Users don't have to decide on or invent their own approach ▪ Enables code optimizations to be made based on the convention

This enhancement provides array like functionality for XML. In version 6.1 there was nothing done to provide array like support for XML, which has sequences but not arrays. A sequence in XML is an element with a maxOccurs attribute greater than one. Multiple elements can exist, but they are a sequence rather than an array in that there can't be any missing elements as you can have with a sparse array. However, using XML it is possible for array like functionality to be used. This is done with a variety of different conventions for defining and manipulating the XML, but there is not one common convention that everyone uses. With the introduction of the array functionality in version 6.2, a convention has been adopted for use. Specific information on how this is done is documented in the Information Center topic entitled "Arrays in business objects". Among the benefits of this enhancement is that implementations can now make use of a common convention. This enables the server code to make optimizations based on the convention. Also, as a user, you can also make use of the same convention without having to research and choose between differing options for enabling array support.



IBM Software Group

Summary

- SCA enhancements presented
  - SCA binding support for Web services feature pack
  - Dynamic invocation for all binding types
  - Consistent fault handling across bindings
  - Support MQ headers for CICS and IMS
- Business data enhancements presented
  - XML validation error reporting
  - Create functionality for set methods
  - Array functionality

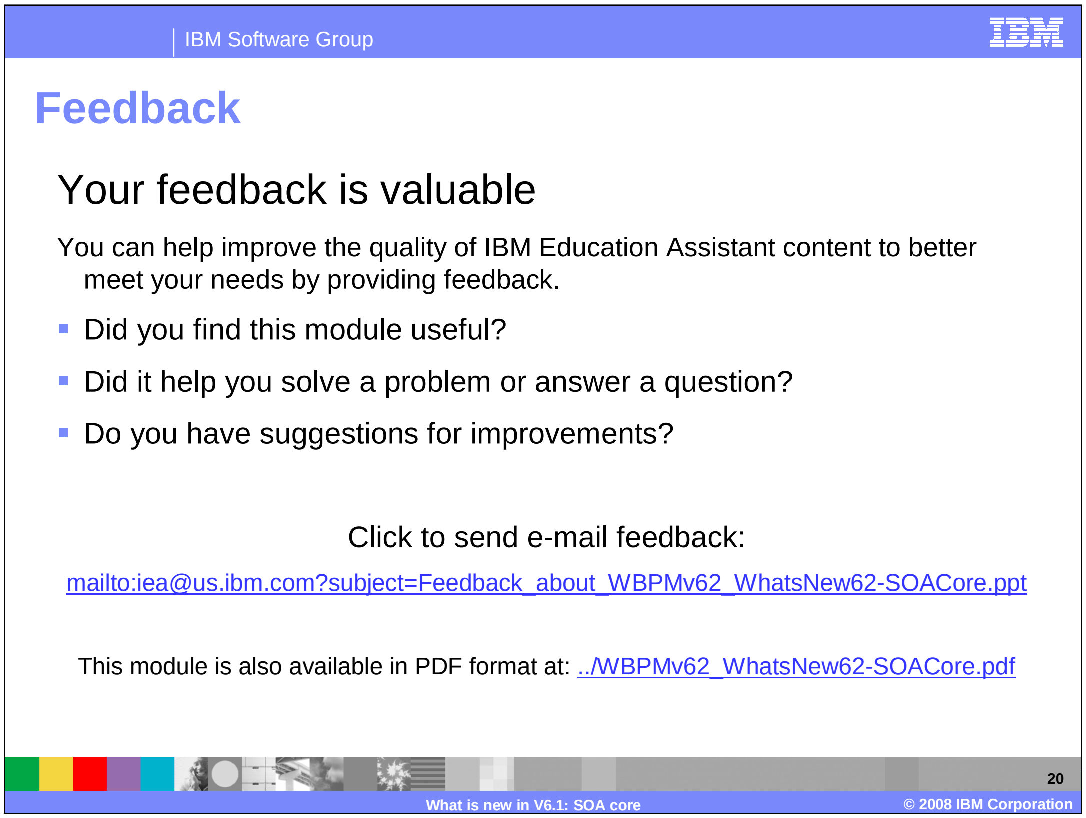
19

What is new in V6.1: SOA core

© 2008 IBM Corporation

In this presentation you saw an overview of the enhancements made in version 6.2 for the SOA core layer of the WebSphere Process Server and WebSphere Enterprise Service Bus. First you were introduced to the enhancements for service component architecture. The most significant of these is the addition of SCA bindings that support the Web services feature pack, which implements support for JAX-WS Web services. The next was the support for dynamic invocation across all binding types, including messaging and HTTP bindings. Fault handling enhancements were described which provide a consistent approach that works across the various types of bindings, with support for both business and server runtime exceptions. The last of the SCA enhancements described was the ability for an import to be configured to statically set values into MQ headers for interaction with CICS and IMS.

The next set of enhancements related to the handling of business objects. A consistent mechanism has been implemented for reporting XML validation errors with error logging that specifically identifies the error encountered. The DataObject APIs have been enhanced to improve the usability of the set methods when used with complex nested business objects. Finally, an approach to providing array like functionality for XML has been adopted.



IBM Software Group

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:

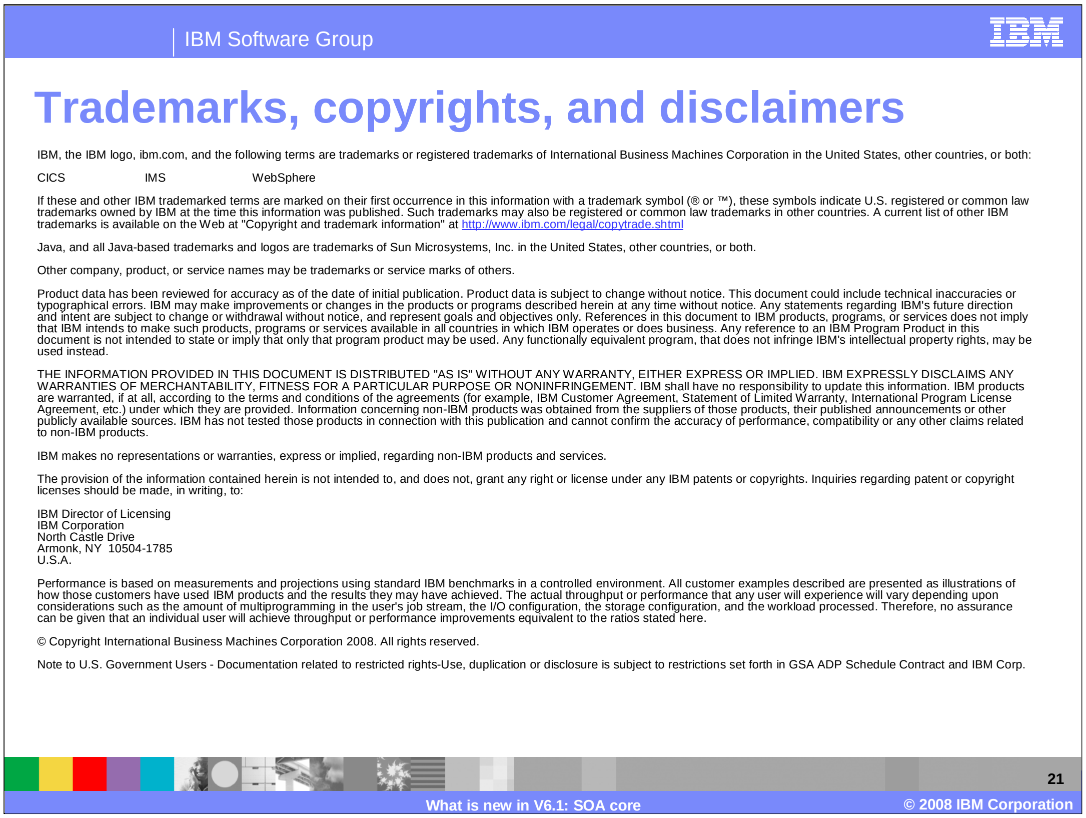
mailto:iea@us.ibm.com?subject=Feedback_about_WBPMv62_WhatsNew62-SOACore.ppt

This module is also available in PDF format at: ../WBPMv62_WhatsNew62-SOACore.pdf

20

What is new in V6.1: SOA core

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



Trademarks, copyrights, and disclaimers

IBM, the IBM logo, ibm.com, and the following terms are trademarks or registered trademarks of International Business Machines Corporation in the United States, other countries, or both:

CICS IMS WebSphere

If these and other IBM trademarked terms are marked on their first occurrence in this information with a trademark symbol (® or ™), these symbols indicate U.S. registered or common law trademarks owned by IBM at the time this information was published. Such trademarks may also be registered or common law trademarks in other countries. A current list of other IBM trademarks is available on the Web at "Copyright and trademark information" at http://www.ibm.com/legal/copytrade.shtml

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

Other company, product, or 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 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.

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.