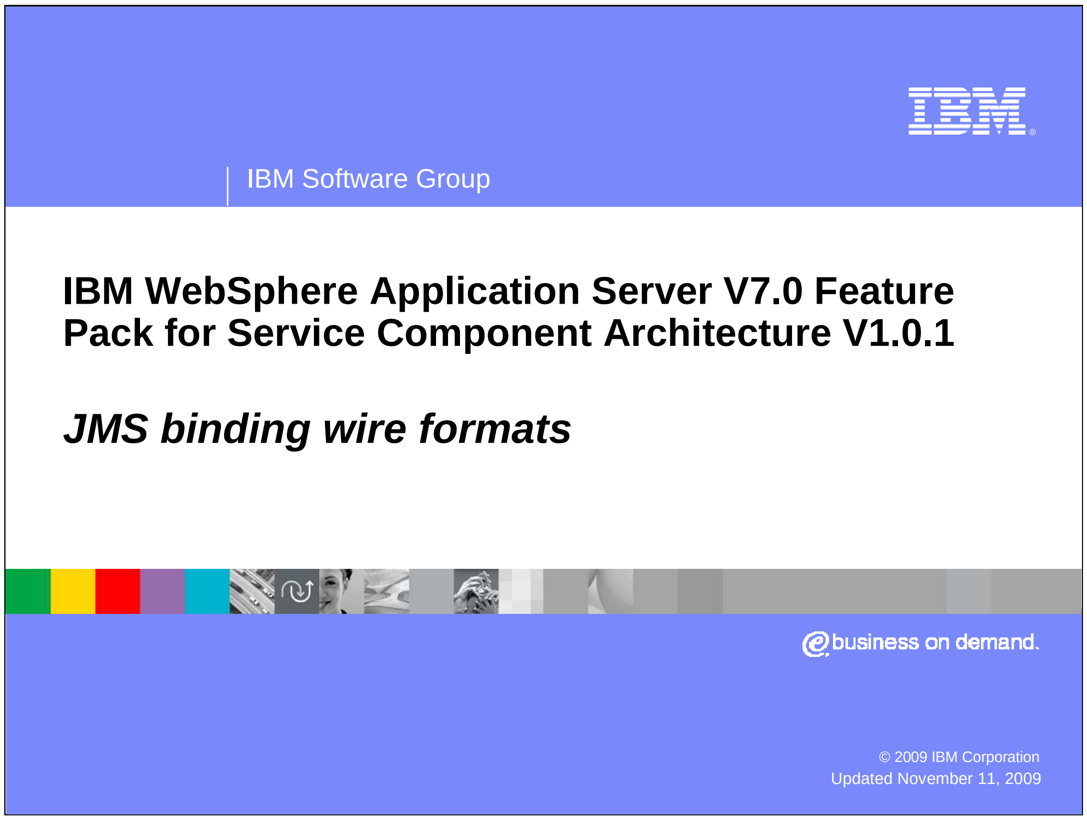
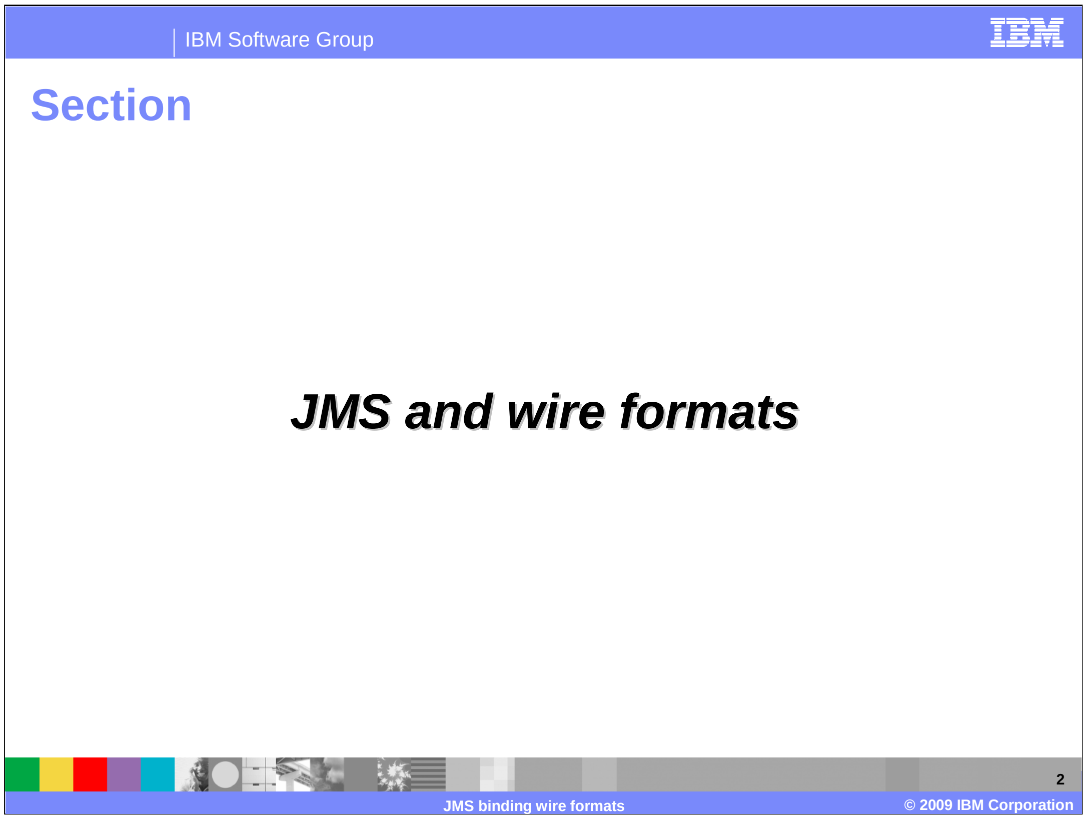


IBM Software Group

IBM WebSphere Application Server V7.0 Feature Pack for Service Component Architecture V1.0.1

JMS binding wire formats

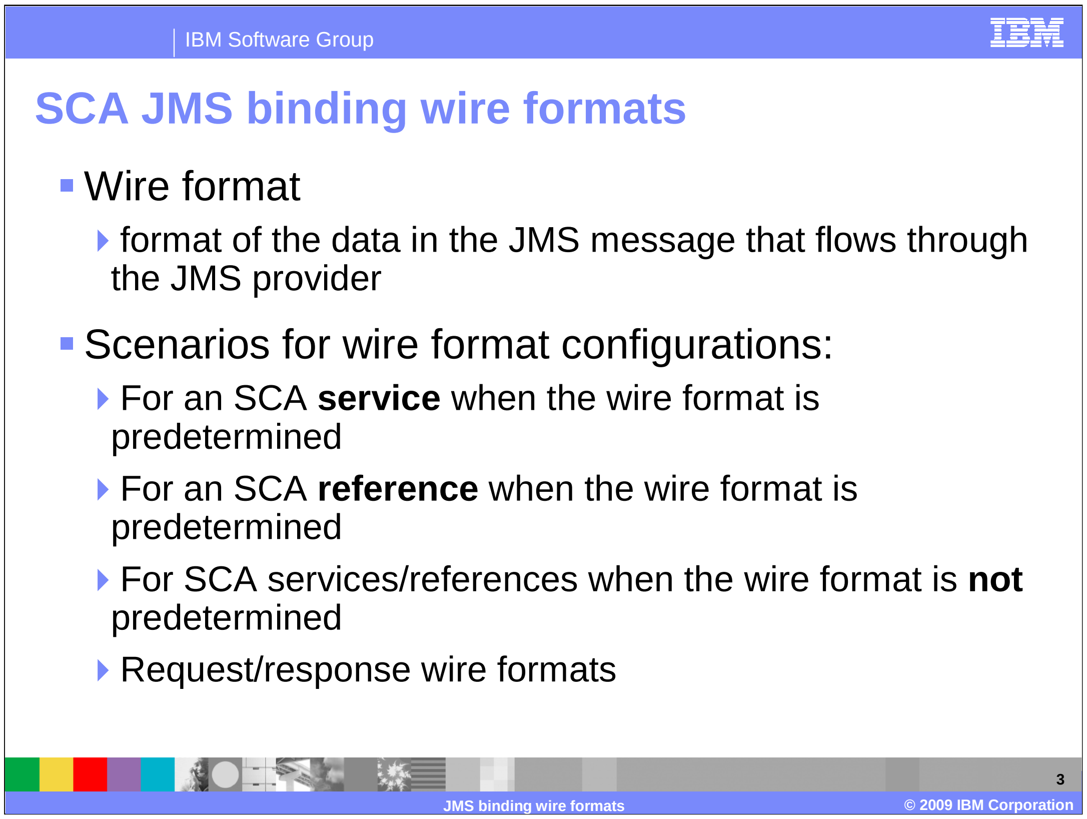
This presentation will discuss JMS binding wire formats.



Section

JMS and wire formats

This section will look at JMS and wire formats.



SCA JMS binding wire formats

- Wire format
  - format of the data in the JMS message that flows through the JMS provider
- Scenarios for wire format configurations:
  - For an SCA **service** when the wire format is predetermined
  - For an SCA **reference** when the wire format is predetermined
  - For SCA services/references when the wire format is **not** predetermined
  - Request/response wire formats

Wire format describes the format of the data that is on the wire. For the SCA JMS binding, the wire format is the format of the data in the JMS message that flows through the JMS provider. Because of the variety of message types and formats, SCA services and references that are configured with a JMS binding might require additional configuration. This is to enable the runtime environment to perform the marshalling and unmarshalling required to translate between application data formats and the format of the JMS message on the wire. The additional configuration of message types is the specification of the wire format for message handling.

Before you configure the wire format, JMS binding on an SCA application has to be already enabled.

When configuring an SCA service or reference, it is important to recognize whether the wire format is previously established by your existing messaging application infrastructure. Also, it is important to recognize whether you are selecting the wire format along with your SCA application. If you are starting with an application with a preexisting messaging infrastructure and you are adding your SCA application to this environment, the wire format is likely already determined by the messaging infrastructure. If you are starting with an SCA application and you intend for this application to interact with JMS message producers or consumers, you can specify the wire format within your SCA application.

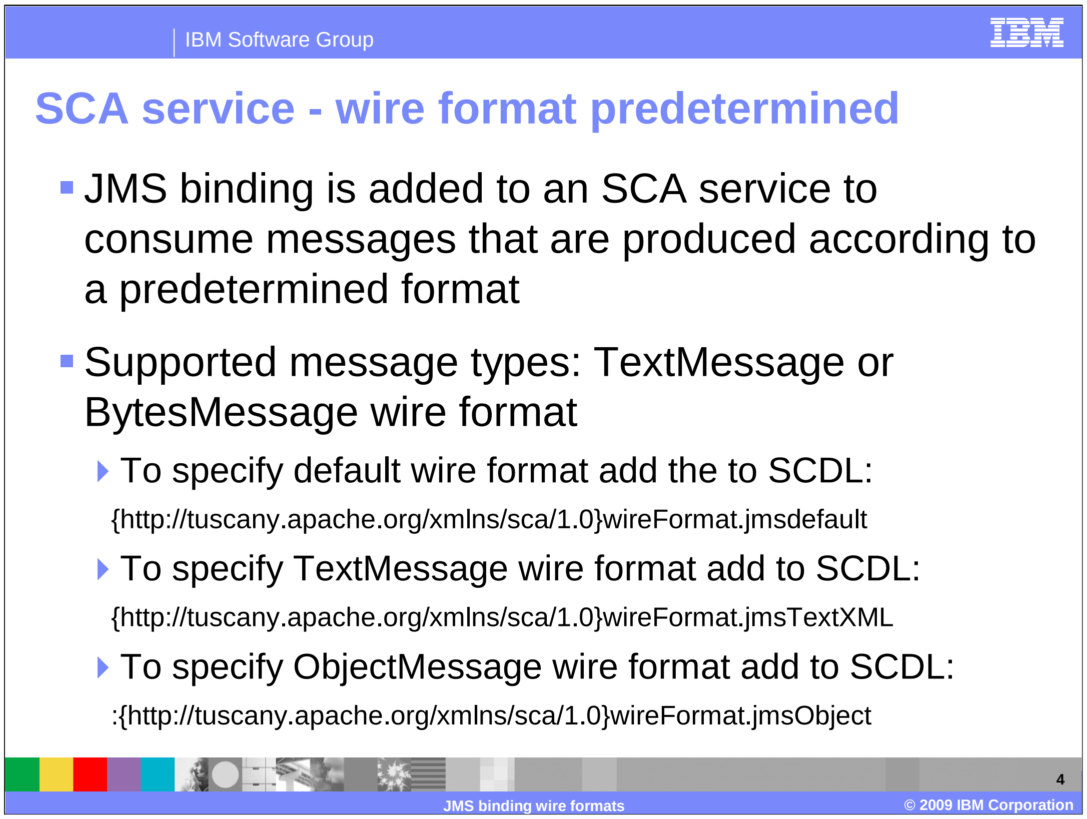
Scenarios for configuring wire formats include:

Configuring the JMS binding for an SCA service when the wire format is predetermined.

Configuring the JMS binding for an SCA reference when the wire format is predetermined.

Configuring request and response wire formats.

Configuring the JMS binding for an SCA services and reference when the wire format is not predetermined.



SCA service - wire format predetermined

- JMS binding is added to an SCA service to consume messages that are produced according to a predetermined format
- Supported message types: TextMessage or BytesMessage wire format
  - To specify default wire format add the to SCDL:

    {http://tuscany.apache.org/xmlns/sca/1.0}wireFormat.jmsdefault
  - To specify TextMessage wire format add to SCDL:

    {http://tuscany.apache.org/xmlns/sca/1.0}wireFormat.jmsTextXML
  - To specify ObjectMessage wire format add to SCDL:

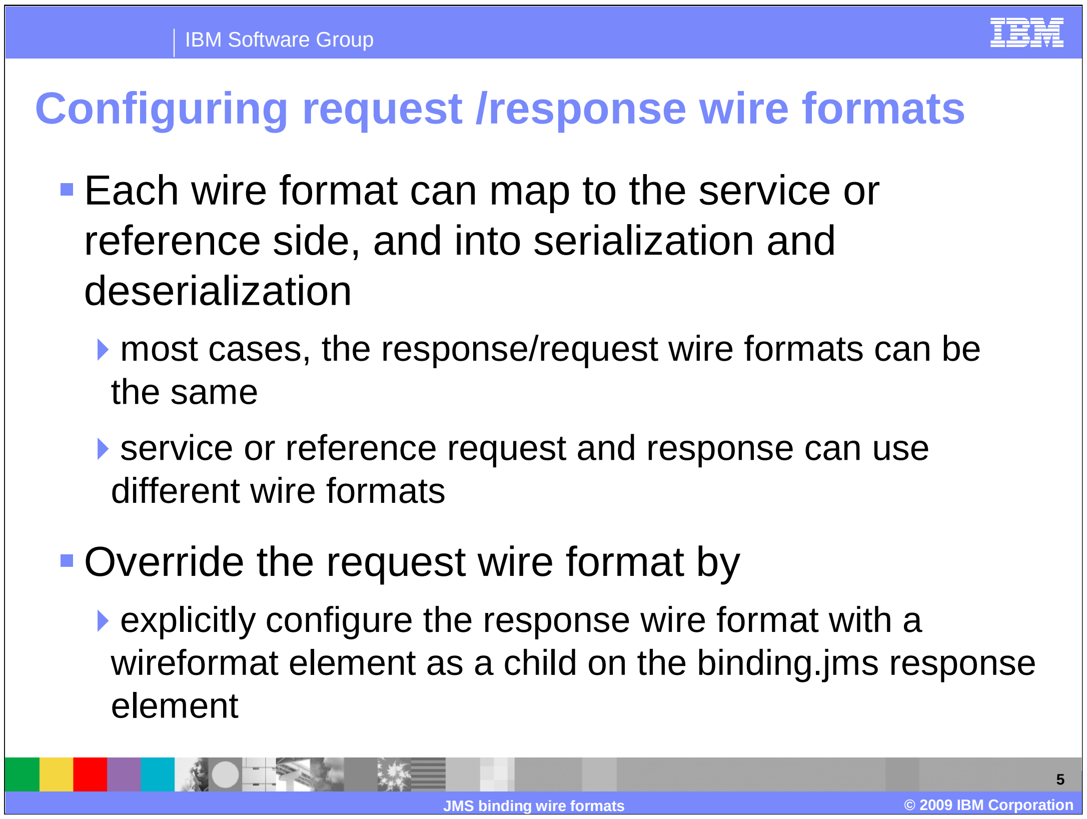
    :{http://tuscany.apache.org/xmlns/sca/1.0}wireFormat.jmsObject

In this scenario, you are adding the JMS binding to an SCA service to consume messages that are produced according to a predetermined format. The supported message types are described for this scenario.

The TextMessage or BytesMessage wire format uses JAXB technology to marshal and unmarshal data into XML. This wire format is the default wire format. Thus, this wire format applies if no wire format element is specified in the composite definition file. To specify this wire format, add this wire format element to the composite definition file:{http://tuscany.apache.org/xmlns/sca/1.0}wireFormat.jmsdefault.

To specify specifically TextMessage wire format add this to the SCDL; {http://tuscany.apache.org/xmlns/sca/1.0}wireFormat.jmsTextXML.

The ObjectMessage wire format uses serialized Java™ objects. It maps to **the java.lang.Object.class.** To specify this wire format, add this wire format element to the composition definition file:{http://tuscany.apache.org/xmlns/sca/1.0}wireFormat.jmsObject.

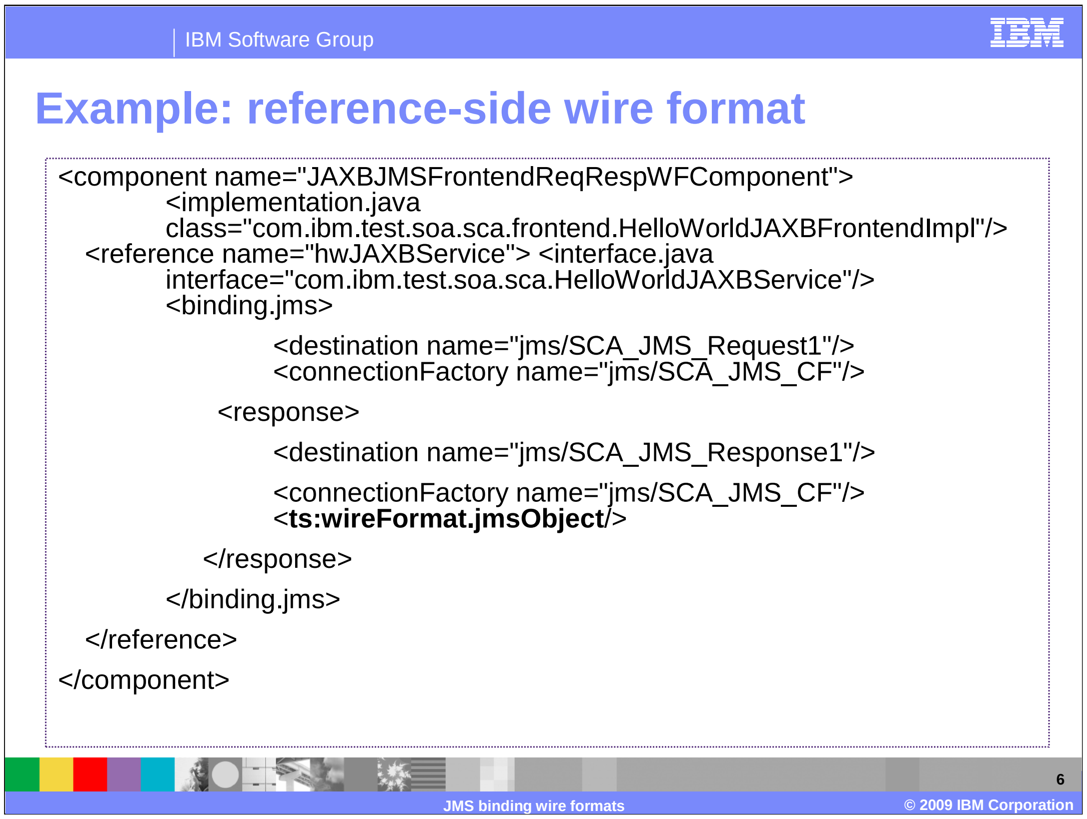


Configuring request /response wire formats

- Each wire format can map to the service or reference side, and into serialization and deserialization
  - most cases, the response/request wire formats can be the same
  - service or reference request and response can use different wire formats
- Override the request wire format by
  - explicitly configure the response wire format with a wireformat element as a child on the binding.jms response element

In general, each wire format can map to the service or reference side, and even into serialization and deserialization. As a result, you can configure each service or reference request and response to use different wire formats.

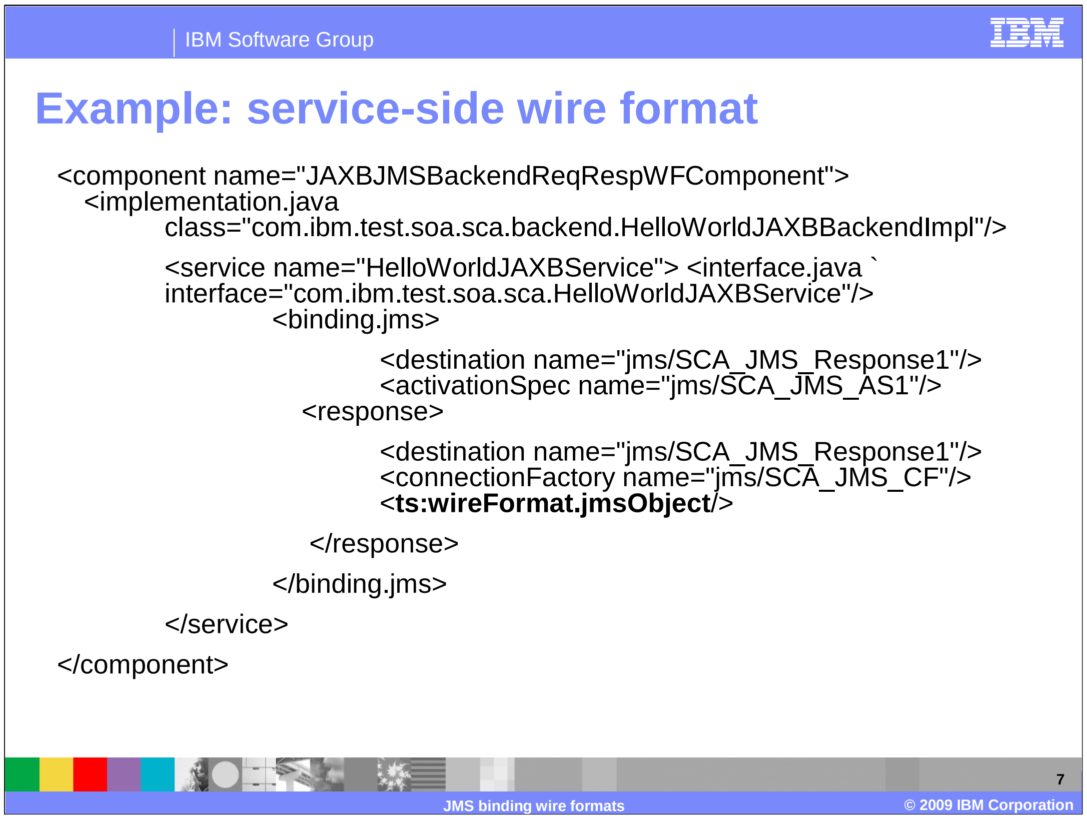
In most cases, the response wire format can be the same as the request wire format for a messaging application. However, in certain scenarios this might not be reasonable, such as when the inputs and outputs of an operation cannot use the same wire format. In this situation, you can override the request wire format by explicitly configuring the response wire format with a wireformat element as a child on the binding.jms response element.



Example: reference-side wire format

```
<component name="JAXBJMSFrontendReqRespWFComponent">
        <implementation.java
        class="com.ibm.test.soa.sca.frontend.HelloWorldJAXBFrontendImpl"/>
  <reference name="hwJAXBService"> <interface.java
        interface="com.ibm.test.soa.sca.HelloWorldJAXBService"/>
        <binding.jms>
                <destination name="jms/SCA_JMS_Request1"/>
                <connectionFactory name="jms/SCA_JMS_CF"/>
            <response>
                <destination name="jms/SCA_JMS_Response1"/>
                <connectionFactory name="jms/SCA_JMS_CF"/>
                <ts:wireFormat.jmsObject/>
           </response>
        </binding.jms>
  </reference>
</component>
```

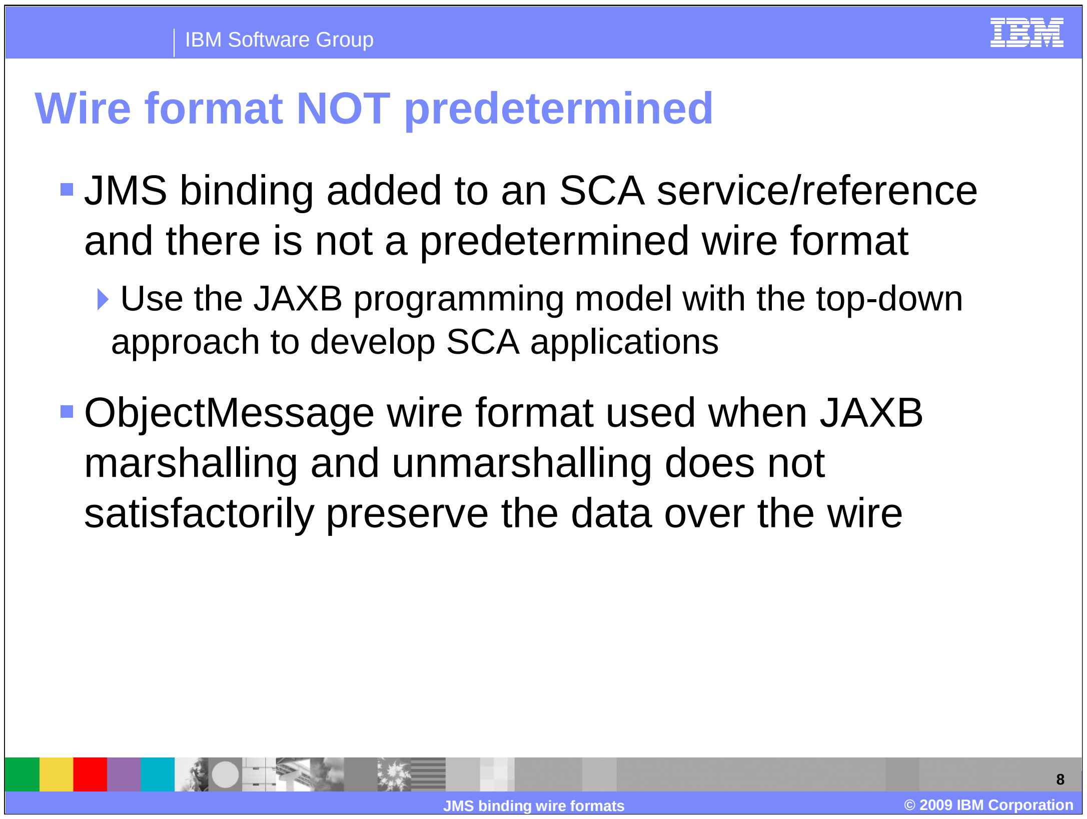
In the above component configuration example, the binding level wire format is the default because no wire format is specified. However, the response wire format is overridden by the jmsObject wire format.



Example: service-side wire format

```
<component name="JAXBJMSBackendReqRespWFComponent">
  <implementation.java
        class="com.ibm.test.soa.sca.backend.HelloWorldJAXBBackendImpl"/>
        <service name="HelloWorldJAXBService"> <interface.java `
        interface="com.ibm.test.soa.sca.HelloWorldJAXBService"/>
                <binding.jms>
                        <destination name="jms/SCA_JMS_Response1"/>
                        <activationSpec name="jms/SCA_JMS_AS1"/>
                  <response>
                        <destination name="jms/SCA_JMS_Response1"/>
                        <connectionFactory name="jms/SCA_JMS_CF"/>
                        <ts:wireFormat.jmsObject/>
                   </response>
                </binding.jms>
        </service>
</component>
```

After you configure the reference-side wire formats, similarly configure the service-side wire format. Here is an example of what the service-side format looks like.

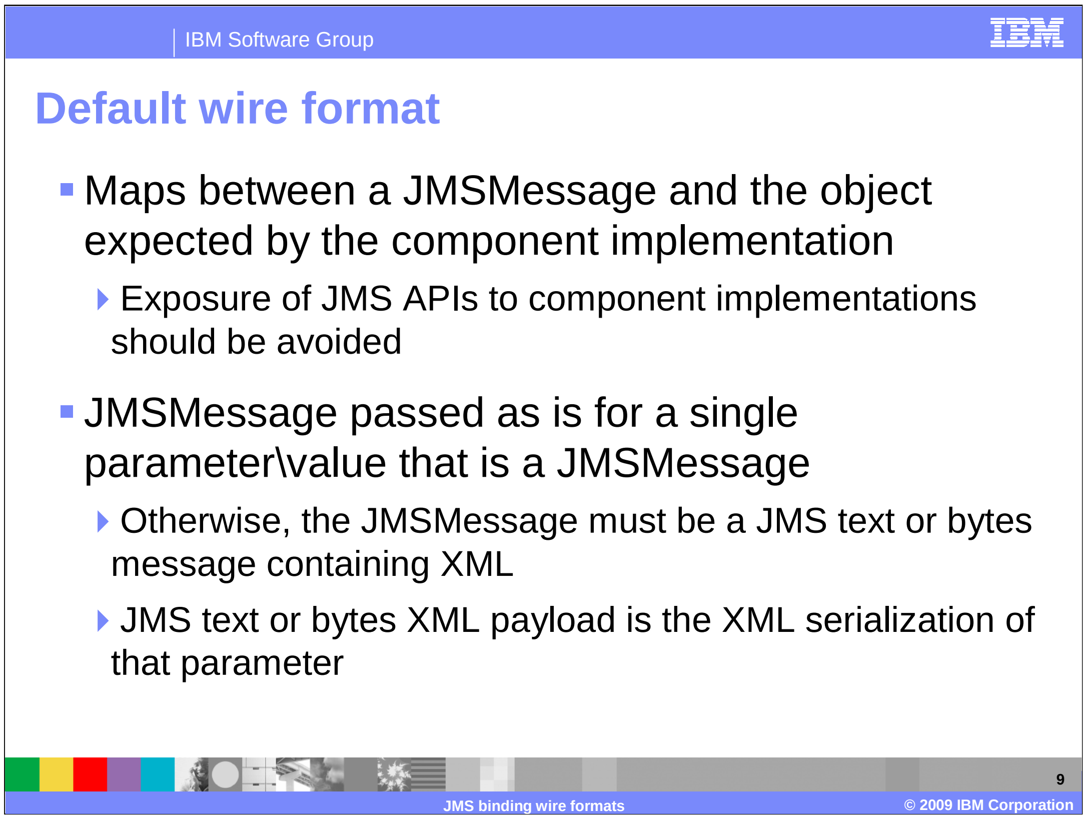


Wire format NOT predetermined

- JMS binding added to an SCA service/reference and there is not a predetermined wire format
  - Use the JAXB programming model with the top-down approach to develop SCA applications
- ObjectMessage wire format used when JAXB marshalling and unmarshalling does not satisfactorily preserve the data over the wire

Wire format is not predetermined. In this scenario, you are adding the JMS binding to an SCA service or reference to produce messages that are consumed by a JMS producer or consumer, and there is not a predetermined wire format. It is a best practice to use the default wire format when starting with the SCA application. Use the JAXB programming model with the top-down approach to developing SCA applications as these service implementations and clients are easily used with the SCA default binding, the SCA Web service binding, and the SCA JMS binding. Adopting an XML-centric view of your business data provides maximum portability across diverse platforms and technologies, and takes advantage of the design goals of a typical SOA environment.

If you have business data that is described within Java classes that implement the Java serialization interface, java.io.Serializable, you can use the ObjectMessage wire format. This is in the scenario where JAXB marshalling and unmarshalling does not satisfactorily preserve the data over the wire.



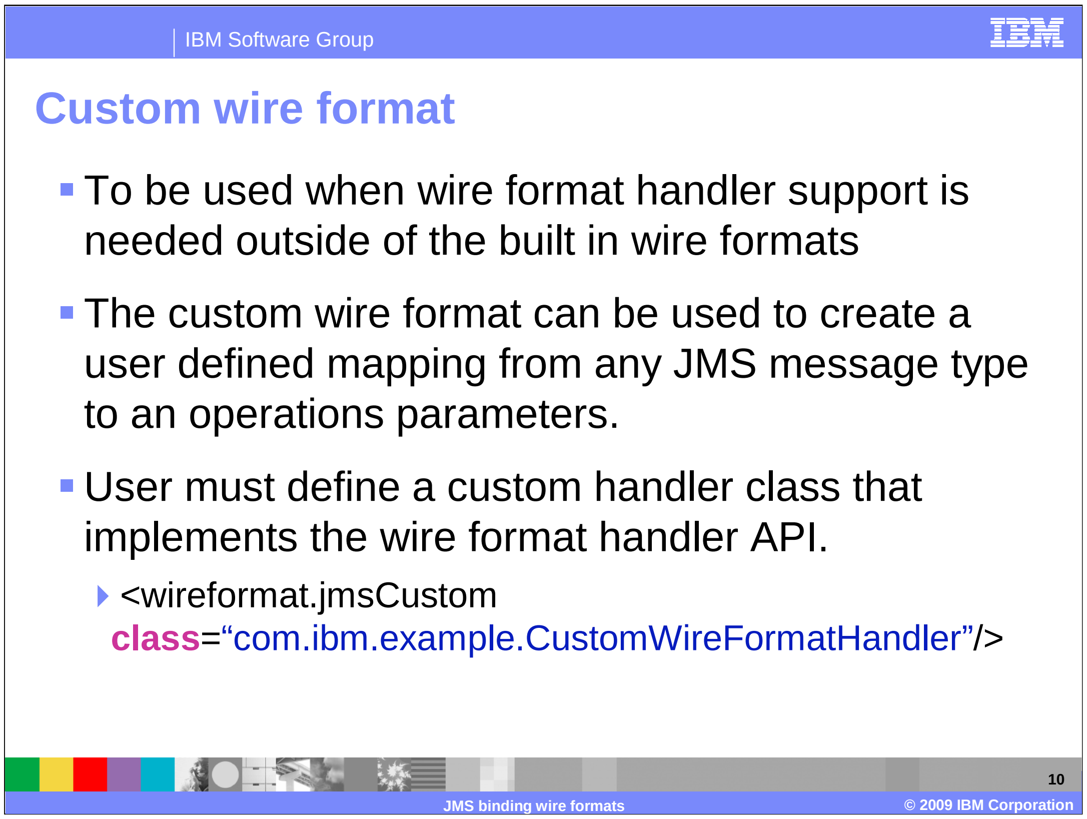
Default wire format

- Maps between a JMSMessage and the object expected by the component implementation
  - Exposure of JMS APIs to component implementations should be avoided
- JMSMessage passed as is for a single parameter\value that is a JMSMessage
  - Otherwise, the JMSMessage must be a JMS text or bytes message containing XML
  - JMS text or bytes XML payload is the XML serialization of that parameter

The default data binding behavior maps between a JMSMessage and the object expected by the component implementation. Component implementers are encouraged to avoid exposure of JMS APIs to component implementations, however in the case of an existing implementation that expects a JMSMessage, this provides for simple reuse of that as an SCA component.

The message body is mapped to the parameters or return value of the target operation if there is a single parameter or return value that is a JMSMessage, then the JMSMessage is passed as is. Otherwise, the JMSMessage must be a JMS text or bytes message containing XML.

If there is a single parameter, or for the return value, the JMS text XML payload is the XML serialization of that parameter according to the WSDL schema for the message. If there are multiple parameters, then they are encoded in XML using the document wrapped style, according to the WSDL schema for the message.

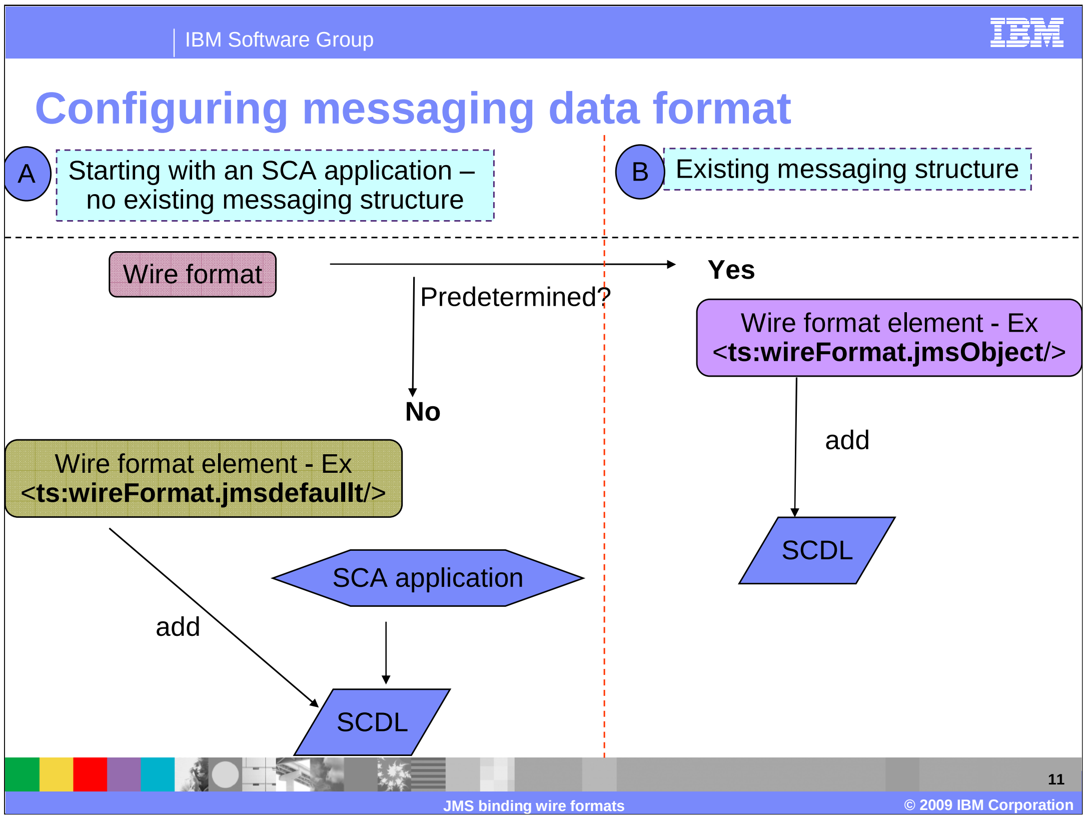


Custom wire format

- To be used when wire format handler support is needed outside of the built in wire formats
- The custom wire format can be used to create a user defined mapping from any JMS message type to an operations parameters.
- User must define a custom handler class that implements the wire format handler API.
  - <wireformat.jmsCustom class="com.ibm.example.CustomWireFormatHandler"/>

Custom wire format is used when wire format handler support is needed outside of the built in wire formats.

The custom wire format can be used to create a user defined mapping from any JMS message type to an operations parameters. With this type of format, user must define a custom handler class that implements the wire format handler API as shown.


IBM Software Group
Configuring messaging data format
A
Starting with an SCA application – no existing messaging structure
B
Existing messaging structure
Wire format
Predetermined?
Yes
No
Wire format element - Ex <ts:wireFormat.jmsObject/>
add
SCDL
Wire format element - Ex <ts:wireFormat.jmsdefaullt/>
SCA application
add
SCDL
11
JMS binding wire formats
© 2009 IBM Corporation

In general you can configure messaging format under two circumstances. If you are starting with an application with a preexisting messaging infrastructure, as shown above in example B, the wire format is likely already determined by the messaging infrastructure. Note that in this case, you are adding your SCA application to this environment. If you are starting with an SCA application, shown in example A, and you intend for this application to interact with some JMS message producers or consumers, you can specify the wire format within your SCA application.

General steps for configuring meesaging data format are:

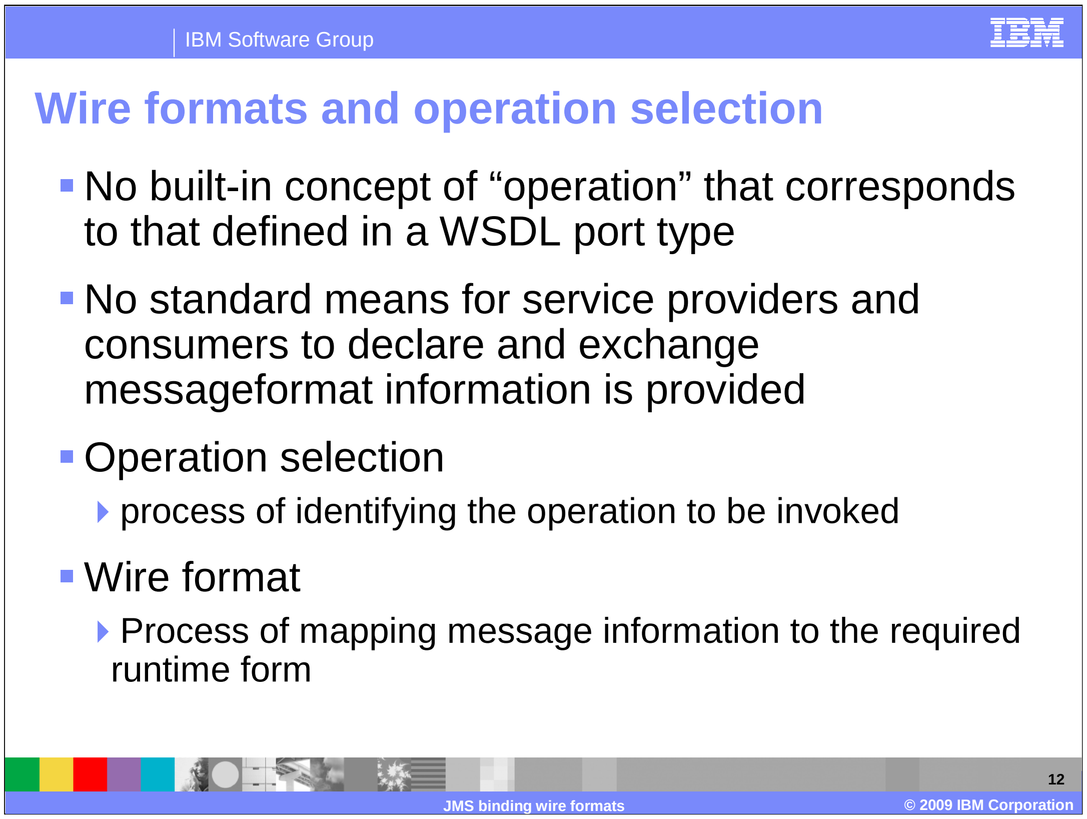
First, determine if you are using a wire format that is predetermined by your existing messaging infrastructure or if you are starting with an SCA application and defining the message wire format.

Second, if you are using a wire format predetermined by your existing messaging infrastructure, add the corresponding wire format element into the composition definition file.

Third, ensure that your SCA service and service client implementation and interfaces map appropriately for the specific wire format that you selected.

Forth, optionally, if you want exception checking to occur over the JMS binding, ensure that the JMS producer and consumer that is interoperating with your SCA application follows the SCA JMS binding exception handling procedures described previously.

Fifth, If you are starting with an SCA application and defining the message infrastructure, add the appropriate wire format element into the composition definition file. Ensure that your JMS producer or consumer applications understand how to interoperate with this message data format.


IBM Software Group

Wire formats and operation selection

- No built-in concept of “operation” that corresponds to that defined in a WSDL port type
- No standard means for service providers and consumers to declare and exchange messageformat information is provided
- Operation selection
  - process of identifying the operation to be invoked
- Wire format
  - Process of mapping message information to the required runtime form

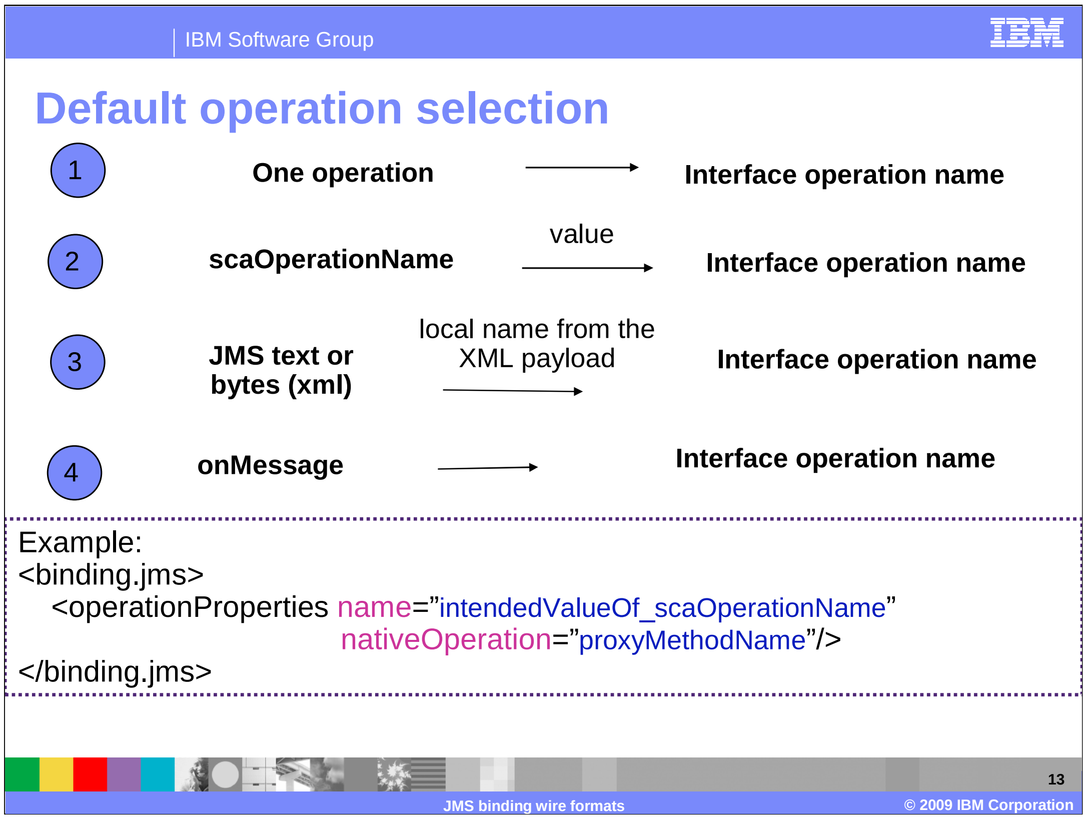
12

JMS binding wire formats

© 2009 IBM Corporation

In general messaging providers deal with message formats and destinations. There is not a typically built-in concept of “operation” that corresponds to that defined in a WSDL port type. Messages have a format which corresponds in some way to the schema of an input or output message of an operation in the interface of a service or reference. However some means is required in order to identify the specific operation and map the message information into the required form.

No standard means for service providers and consumers to declare and exchange messageformat information is provided. The process of identifying the operation to be invoked is ***operation selection***; that of mapping message information to the required runtime form is ***data binding***. The JMS binding defines default operation selection and data binding behavior; SCA providers provide extensions for custom behavior.


IBM Software Group
IBM

Default operation selection

1
One operation
Interface operation name
2
scaOperationName
value
Interface operation name
3
JMS text or bytes (xml)
local name from the XML payload
Interface operation name
4
onMessage
Interface operation name

Example:

```
<binding.jms>
   <operationProperties name="intendedValueOf_scaOperationName"
                        nativeOperation="proxyMethodName"/>
</binding.jms>
```

13
JMS binding wire formats
© 2009 IBM Corporation

The JMS binding defines a String message property called scaOperationName, that is used by the JMS binding to map the message to the intended operation on the service. When receiving a request at a service, or a callback at a reference, the JMS binding uses this algorithm to determine the operation name:

If there is only one operation on the service interface, it is assumed that this operation is the operation name for the request.

Otherwise, if the JMS property scaOperationName is set, the value of this property is used as the operation name.

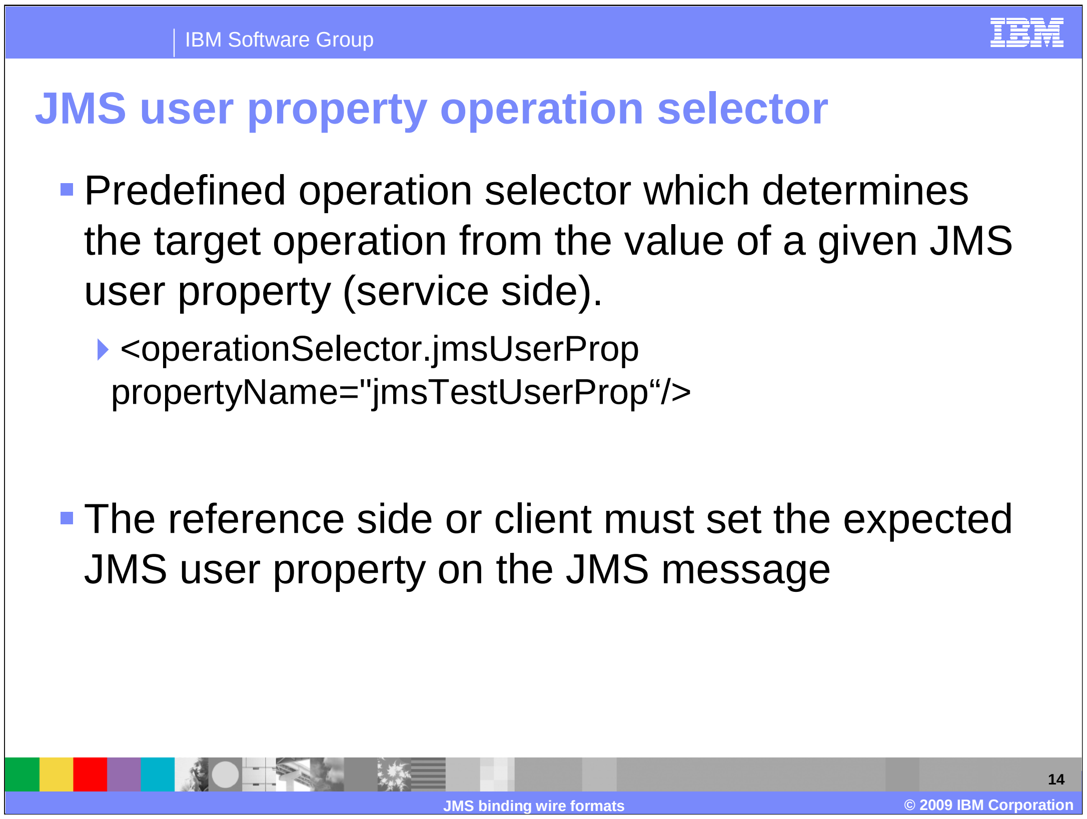
Otherwise, if the message is a JMS text or bytes message containing XML, then the selected operation name is taken from the local name of the root element of the XML payload. This operation selection behavior is only supported over the jmsdefault, jmsTextXML, and jmsBytesXML wire formats.

Otherwise, it is assumed that the operation name is onMessage.

When sending a request from a reference, or a callback from a service, the JMS reference binding sets the scaOperationName message property to the name of the operation that is invoked. By default, this is the name of the operation invoked on the client proxy; however, you can override the operation by using the @nativeOperation attribute within the operationProperties element.

For example:

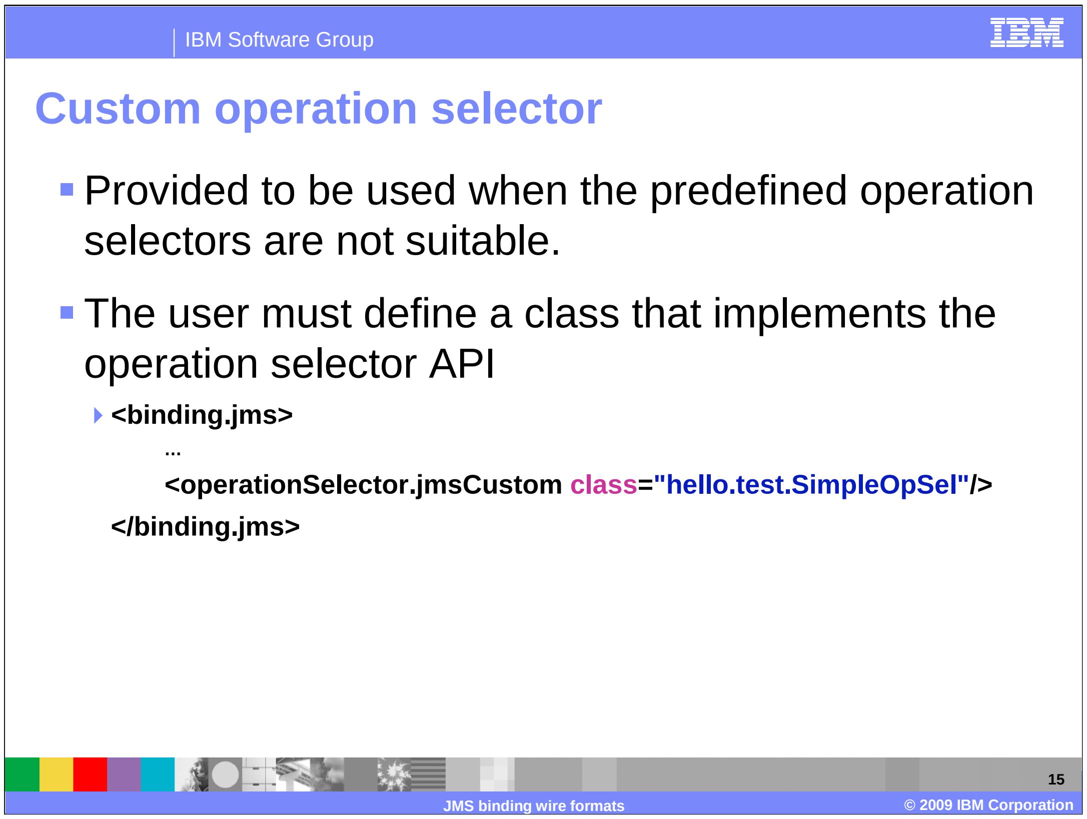
<binding.jms> <operationProperties name="intendedValueOf_scaOperationName" nativeOperation="proxyMethodName"/> </binding.jms>



JMS user property operation selector

- Predefined operation selector which determines the target operation from the value of a given JMS user property (service side).
  - <operationSelector.jmsUserProp propertyName="jmsTestUserProp“/>
- The reference side or client must set the expected JMS user property on the JMS message

JMS User Property operation selector requires a JMS user property to be set to the target operation name. If the property is not set or there is no matching operation it will NOT default to any other value as is the case with the default operation selector.



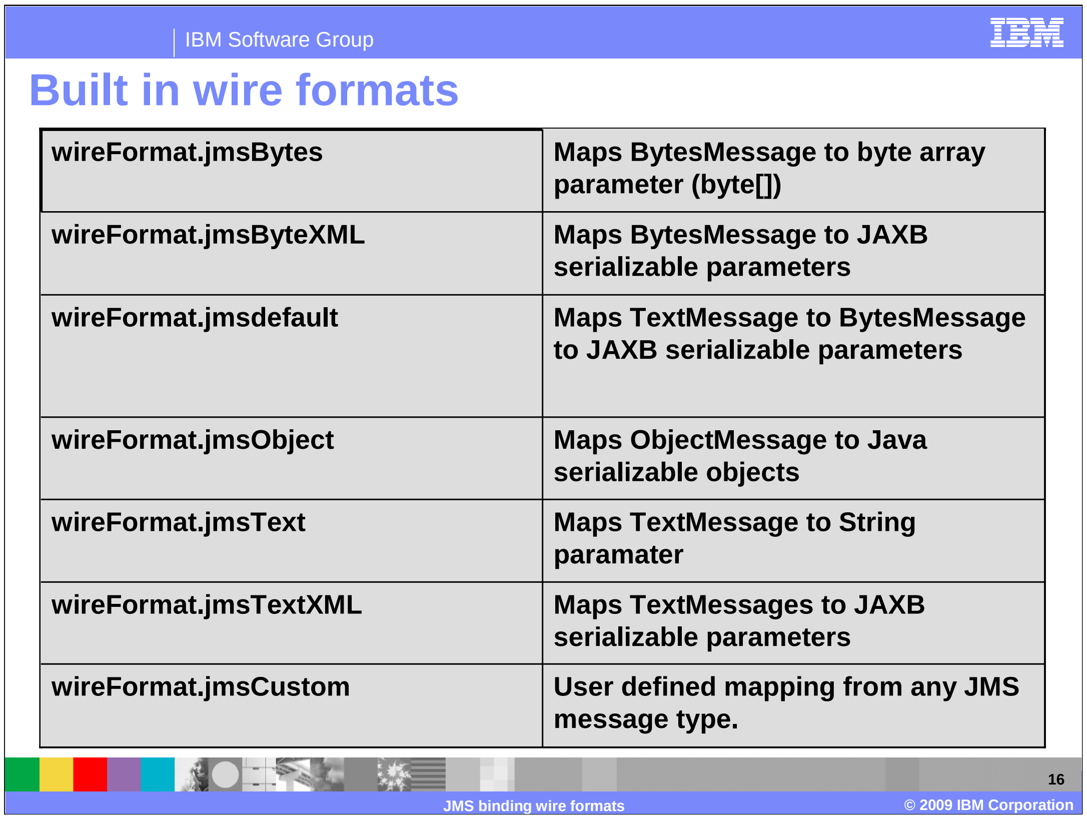
Custom operation selector

- Provided to be used when the predefined operation selectors are not suitable.
- The user must define a class that implements the operation selector API

```
<binding.jms>
      …
      <operationSelector.jmsCustom class="hello.test.SimpleOpSel"/>
</binding.jms>
```

In cases where the predefined operation selectors are not suitable to determine the target operation the ability to provide a customer operation selector has been included. In general, your application only contains business logic. This helps to increase the portability of the application. The custom operation selector provides a way to configure a user defined operation selector through the SCDL without any changes to the application layer.

The operation selector API exposes an instance of javax.jms.Message to you and allows interaction with the JMS user properties and message body.


IBM Software Group

Built in wire formats

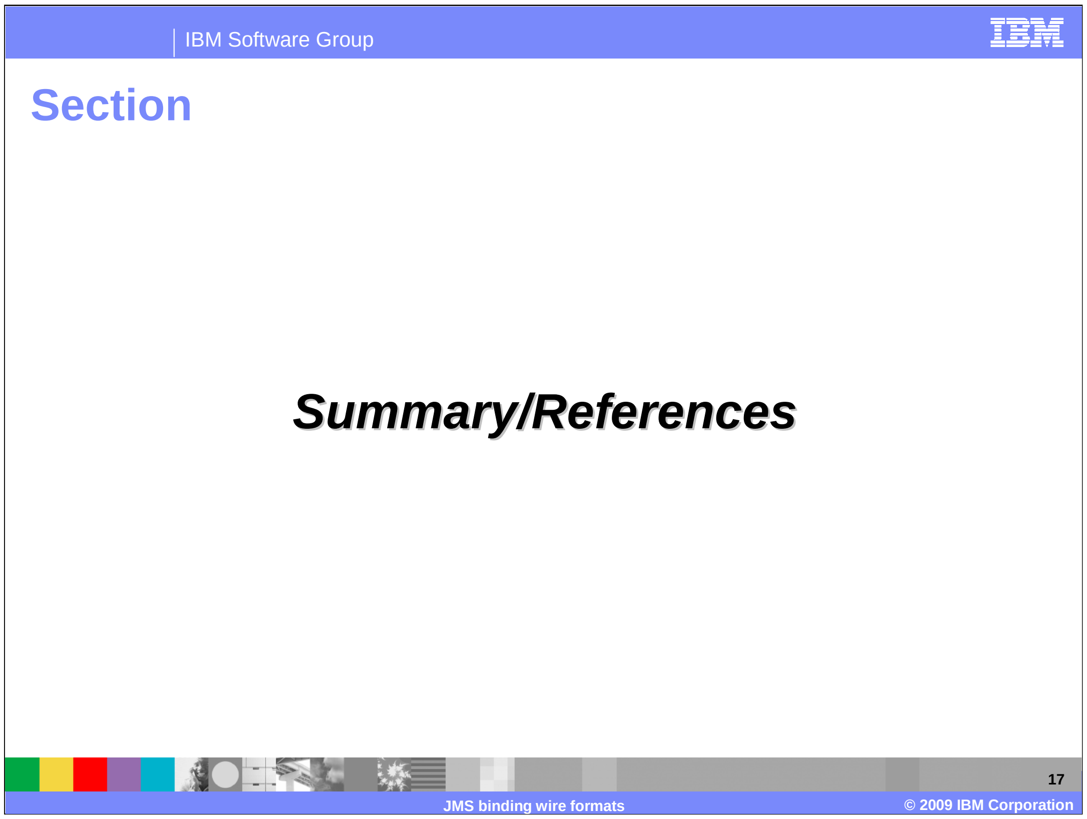
wireFormat.jmsBytes	**Maps BytesMessage to byte array parameter (byte[])**
wireFormat.jmsByteXML	**Maps BytesMessage to JAXB serializable parameters**
wireFormat.jmsdefault	**Maps TextMessage to BytesMessage to JAXB serializable parameters**
wireFormat.jmsObject	**Maps ObjectMessage to Java serializable objects**
wireFormat.jmsText	**Maps TextMessage to String paramater**
wireFormat.jmsTextXML	**Maps TextMessages to JAXB serializable parameters**
wireFormat.jmsCustom	**User defined mapping from any JMS message type.**

16

JMS binding wire formats

© 2009 IBM Corporation

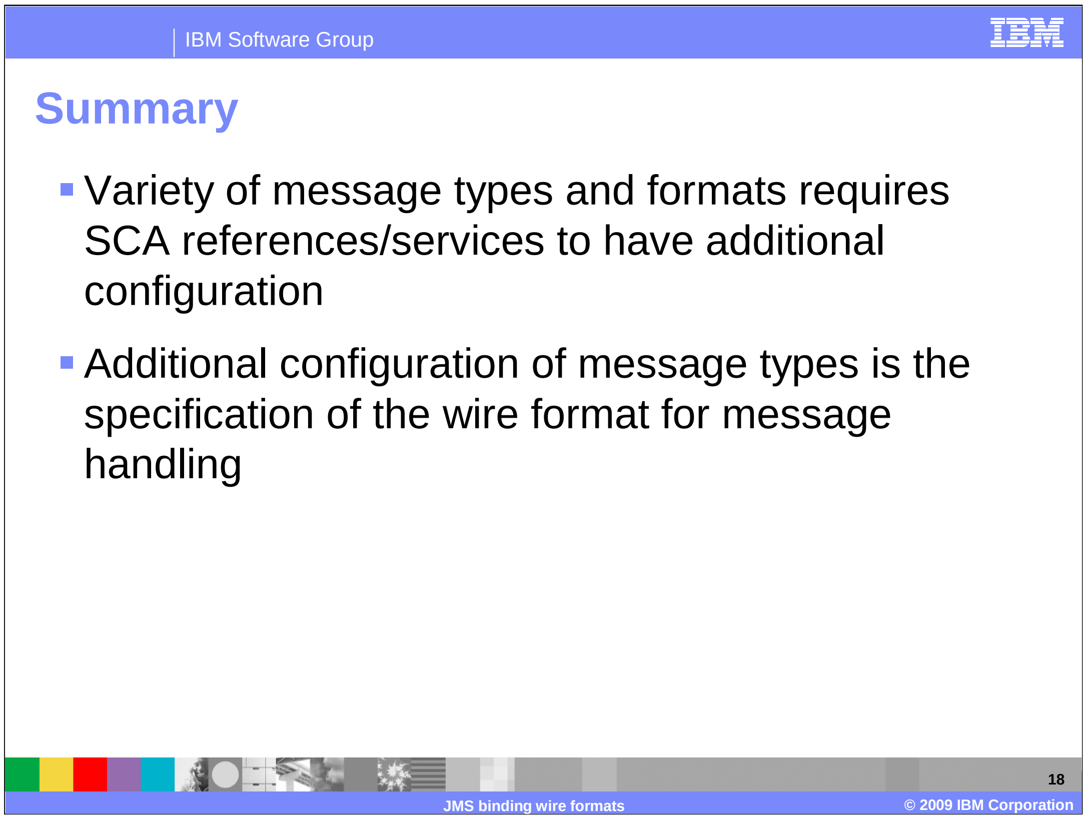
Here is a chart of built in wire formats that you can reference. This is mostly for your own reference.



Section

Summary/References

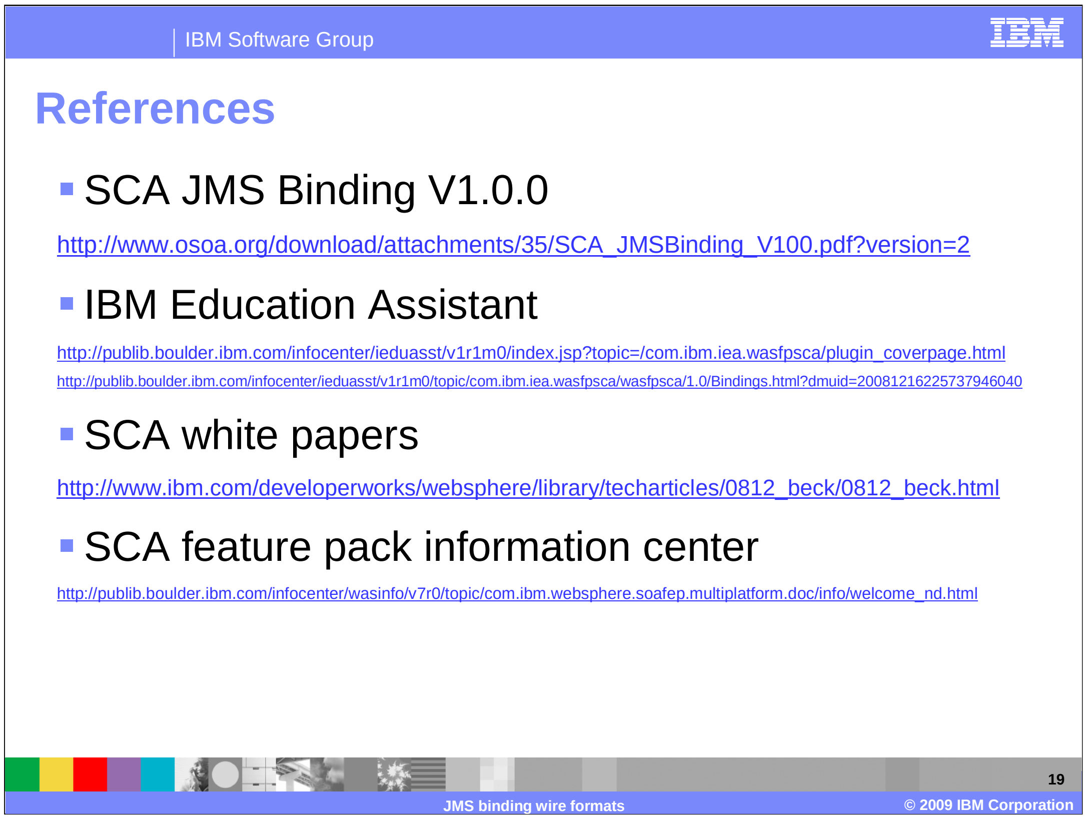
The next section provides a summary and references for this presentation.



Summary

- Variety of message types and formats requires SCA references/services to have additional configuration
- Additional configuration of message types is the specification of the wire format for message handling

SCA services and references that are configured with a JMS binding require additional configuration because of the variety of message types and formats. This is to enable the runtime environment to perform the marshalling and unmarshalling required to translate between application data formats and the format of the JMS message on the wire. The additional configuration of message types is the specification of the wire format for message handling.



References

- SCA JMS Binding V1.0.0

http://www.osoa.org/download/attachments/35/SCA_JMSBinding_V100.pdf?version=2

- IBM Education Assistant

http://publib.boulder.ibm.com/infocenter/ieduasst/v1r1m0/index.jsp?topic=/com.ibm.iea.wasfpsca/plugin_coverpage.html

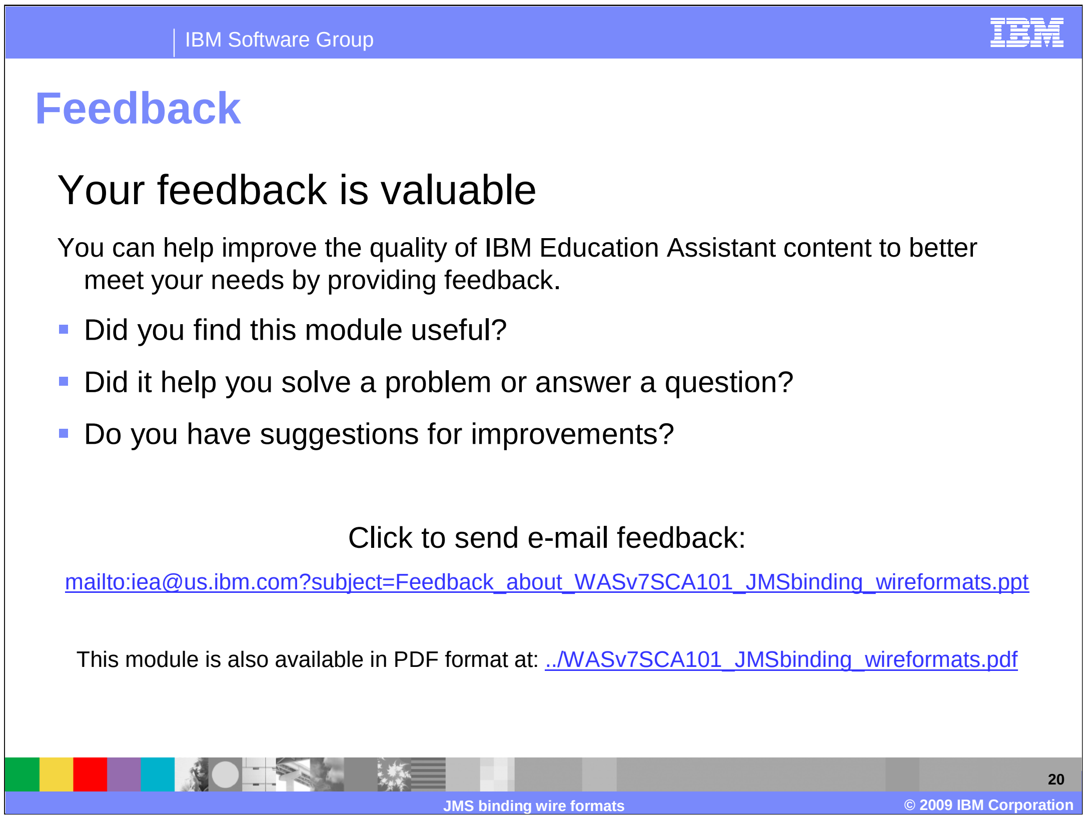
http://publib.boulder.ibm.com/infocenter/ieduasst/v1r1m0/topic/com.ibm.iea.wasfpsca/wasfpsca/1.0/Bindings.html?dmuid=20081216225737946040

- SCA white papers

http://www.ibm.com/developerworks/websphere/library/techarticles/0812_beck/0812_beck.html

- SCA feature pack information center

http://publib.boulder.ibm.com/infocenter/wasinfo/v7r0/topic/com.ibm.websphere.soafep.multiplatform.doc/info/welcome_nd.html



Feedback

Your feedback is valuable

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

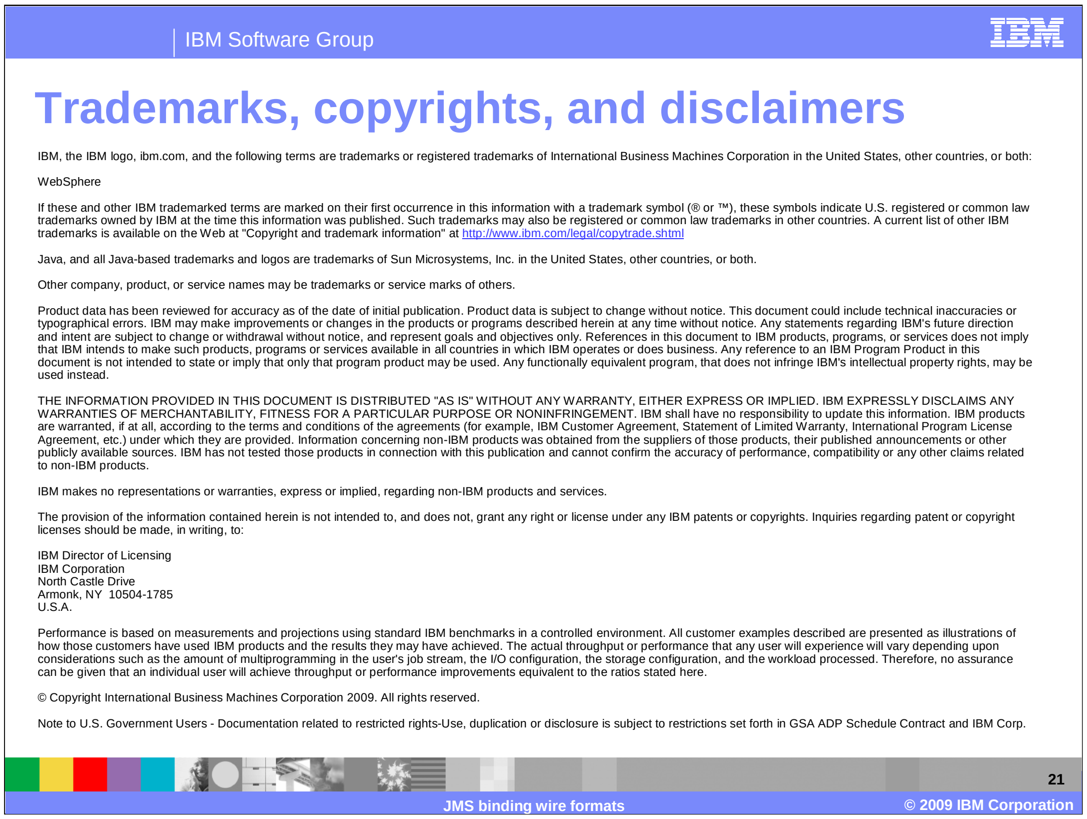
- Did you find this module useful?
- Did it help you solve a problem or answer a question?
- Do you have suggestions for improvements?

Click to send e-mail feedback:

mailto:iea@us.ibm.com?subject=Feedback_about_WASv7SCA101_JMSbinding_wireformats.ppt

This module is also available in PDF format at: ../WASv7SCA101_JMSbinding_wireformats.pdf

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



Trademarks, copyrights, and disclaimers