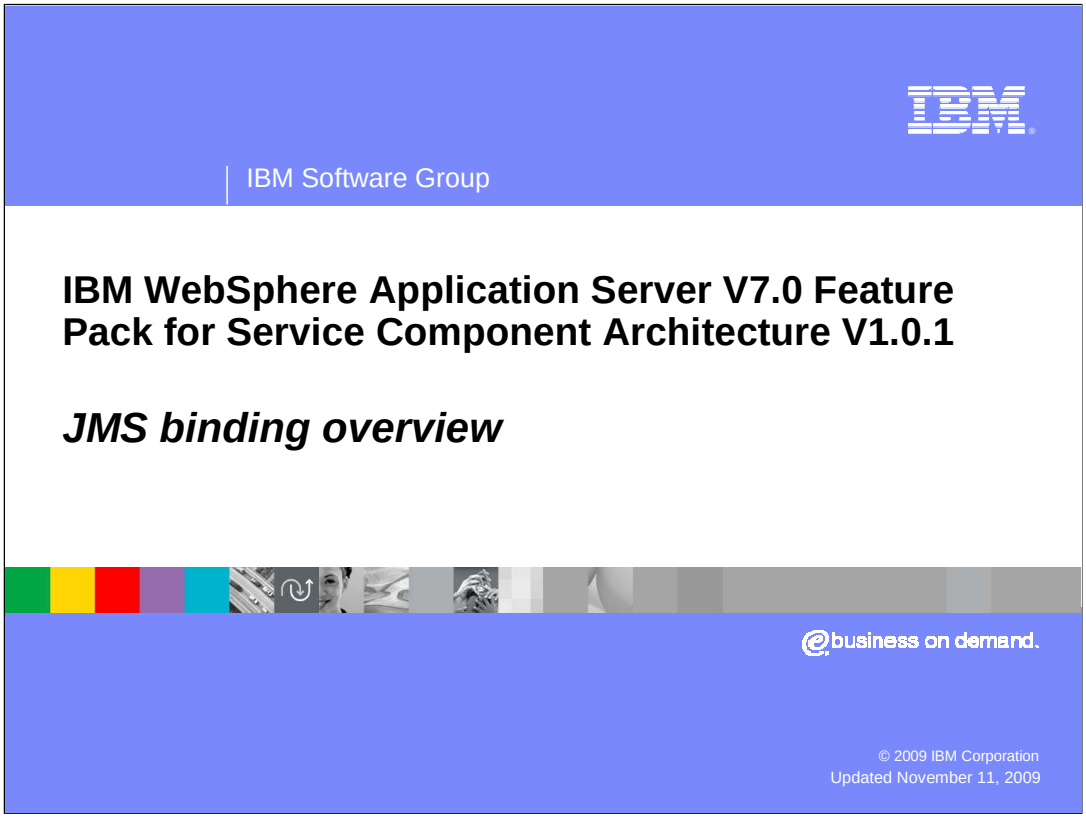


This presentation will discuss Java™ Message Service (JMS) binding overview in Service Component Architecture (SCA).

Section

JMS overview

This section provides a general overview of JMS.

Overview

- JMS represents the Java interfaces for direct connectivity to Messaging Backbones like WebSphere® MQ and WebSphere ESB
- JMS support for SCA is implemented as specified in the JMS binding v1.0.0 specification of OSOA specifications
- Using SCA, new services assembled and composed from new and existing services can be made available over JMS

JMS represents the Java interfaces for direct connectivity to customer's pervasive Enterprise Messaging Backbones like WebSphere MQ and Enterprise Service Buses like WebSphere ESB. It also provides JMS support for the Service Component Architecture JMS Binding V1.00 Specification as documented at OSOA. In addition, using SCA, new services can be made available over JMS creating an open implementation neutral service oriented description of the newly created service assembly and composition. These services can reside on the Enterprise messaging backbone (WebSphere MQ) or enterprise service bus (WebSphere ESB).

Messaging history

- Early “messaging” techniques permitted applications to talk to each other
  - Messages asynchronous and loosely-coupled
  - Language independent (specific APIS implement protocols)
- Message formats only known to applications
- WebSphere MQ introduced in March 1992

Early “messaging” techniques permitted applications to talk to each other. Sender “sends” a message, and receiver “receives” a message. These messages were asynchronous and loosely-coupled. They were also language independent; sender language did not need to match receivers language. Language specific APIs implemented vendor specific protocols. Also, message formats were known only to applications.

Early techniques also allowed sender to send messages to receiver without receiver having to be up and available; sender might not even know who the message is intended for. WebSphere MQ was introduced in March 1992.

Messaging techniques

- Point to point (P2P)
  - One sender/one receiver
  - Operates a Queue
- Publish/Subscribe (pub/sub)
  - One publisher/zero to many receivers
  - Operates against a topic

For those not familiar with messaging, there are two messaging programming styles. First, there is point to Point *(P2P)* that involves one sender and one receiver. This style operates against a “queue resource”. A classic example is a postcard application; to: From:.

The other style is the Publish/Subscribe (Pub/Sub) where there is one publisher and zero to many receivers. This style operates against a topic; for example, li “/sports/scores.” a classic example is a stock market ticker trader that publishes messages to a topic. Speculator's subscribes to well-known topics and receives price update messages.

Another example is where a broker manages subscriptions independently of publisher and the publisher does not know who is subscribed or how many subscriptions there are.

Message types

- JMS messages have a body and associated properties
- JMS messages define the “type” of body that is attached to the message.
  - “Text”
  - “Bytes”
  - “Stream”
  - “Object”
  - “Map”

JMS messages have a body and associated properties. You can think of properties as “headers.

JMS messages define the “type” of body that is attached to the message.
Examples include: “Text”, “Bytes”, “Stream”, “Object”, and “Map.”

Security

- Secure connectivity/authentication is from the sender/receiver to the messaging provider – not between each other
- Authorization is against a given resource (queue, topic) and the given sender/receiver

As far as security is concerned, secure connectivity and authentication is from the sender and receiver to the messaging provider – not between each other. There is no standard to say who the caller is. For JEE “Receivers” it’s really the server/application that is being authenticated. JEE connection administration can be configured with Userid/Password alias. Authorization is against a given resource (queue, topic) and the given sender/receiver.

JMS and Java connector architecture 1.5 (JCA)

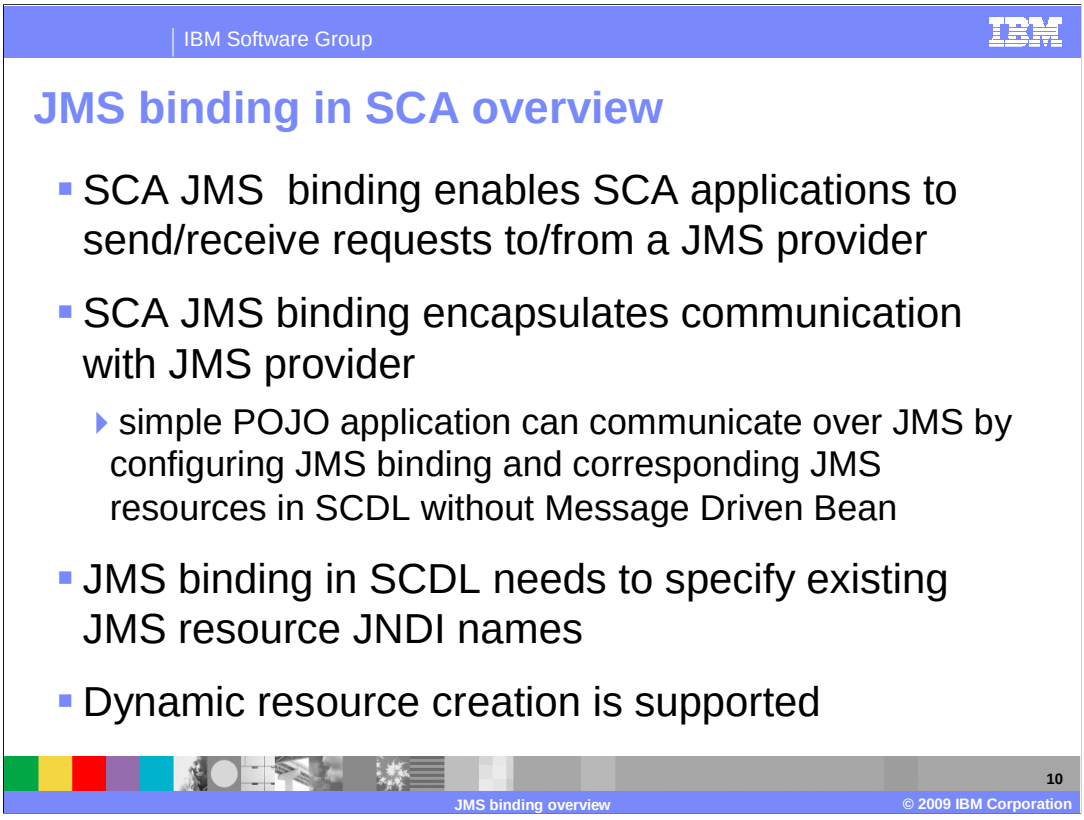
- WebSphere JMS adaptor is a hybrid JCA 1.5 adapter:
  - Special code to provide management of both the JMS connections and sessions
  - Deprecated support for message listener ports
- From SCA standpoint:
  - SCA JMS service requires a JCA 1.5 adapter
  - SCA JMS service does not use an MDB in the background

IBM WebSphere JMS adaptor is a hybrid JCA 1.5 adaptor. It has special code to provide management of both the JMS connections and sessions. There is deprecated support for Message Listener Ports from which support is not provided for with SCA. From an SCA standpoint, an SCA JMS service requires a JCA 1.5 adapter and it works directly with the JCA adapter and does not use an MDB in the background.

Section

JMS in SCA

This next section will discuss JMS in SCA



Using the SCA JMS binding, you can compose and assemble SCA services that are available over JMS or you can use existing JMS applications within an SCA environment. You can use the SCA JMS binding element, <binding.jms>, within either a component service or a component reference definition. When this binding is attached to the component service interface, the JMS binding enables client applications to access an SCA component through a JMS provider. In the case where the JMS binding is used with a component reference, components in an SCA composite can consume an external JMS application and access it just like any other SCA component.

WebSphere Application Server supports asynchronous messaging using JMS. The default messaging provider enables enterprise applications deployed on WebSphere Application Server to perform asynchronous messaging without the need for you to install a JMS provider. The default messaging provider is installed and runs as part of WebSphere Application Server. The feature pack for SCA supports the default messaging provider and MQ as the messaging engine. Keep in mind SCA JMS binding enables SCA applications to send requests to a JMS provider.

SCA JMS support enables SCA applications to receive messages from a JMS provider, thus exposing an SCA service to any JMS client or SCA reference with a JMS binding.

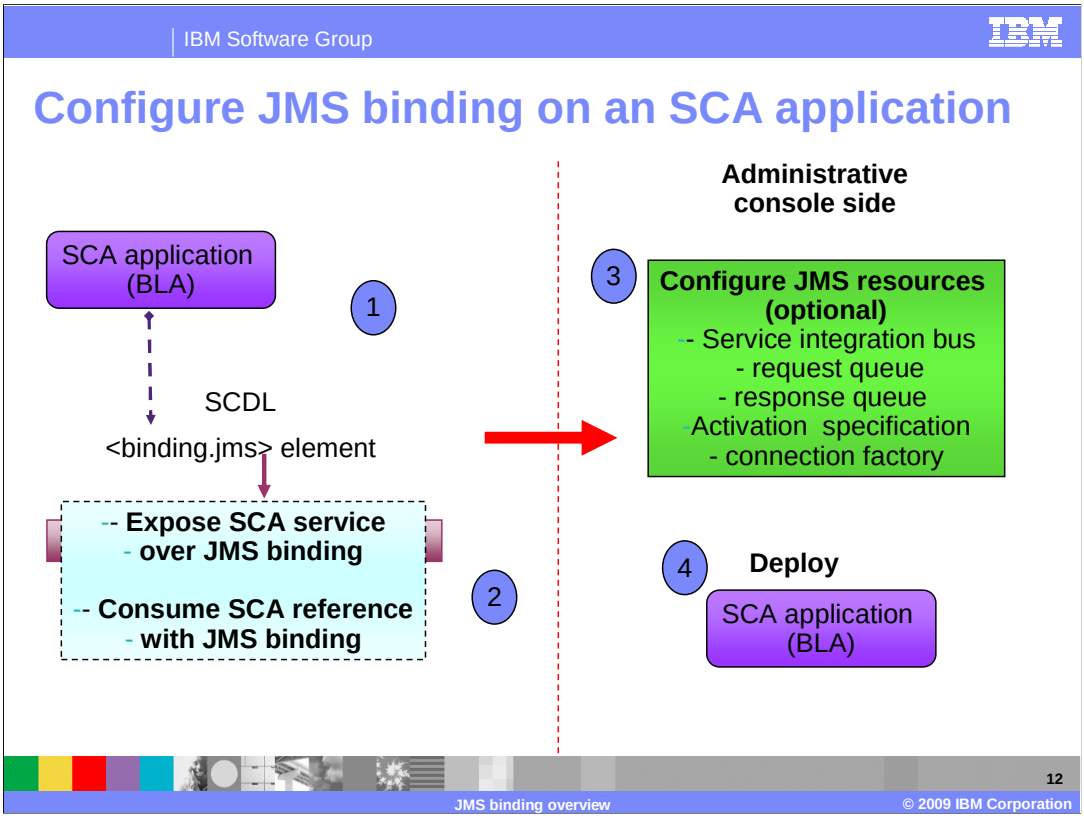
SCA JMS binding encapsulates communication with JMS provider, thus enabling pure POJO SCA application to send and receive messages over JMS. Hence a simple POJO application can communicate over JMS by just configuring JMS binding and corresponding JMS resources in SCDL without Message Driven Bean.

JMS binding in SCDL needs to specify existing JMS resource JNDI names to be able to communicate with JMS providers or let the deployment generate the JMS resources based on component information.

Configuring SCA JMS binding

1. Identify the SCA business-level application (BLA) that you want to enable for JMS messaging
2. Expose SCA service over JMS binding by add <binding.jms> element
   - <activationSpec> to connect to a JMS destination to process request messages
   - <connectionFactory> for request-response
   - <destination> for responses
3. Consume SCA reference with the jms binding
   - <connectionFactory> to identify the JNDI name <destination> for responses
   - <destination> to identify the JMS queue or to
   - Response <destination> describes the JMS destination queue
4. Configure JMS resources (optional)
   - Service integration bus, request/response queues, activation specification, connection factory
5. Deploy SCA application
6. Information center:
   http://publib.boulder.ibm.com/infocenter/wasinfo/beta/topic/com.ibm.websphere.soafep.multiplatform.doc/info/ae/ae/tsca_scajmsbinding.html

Here is an outline of the steps for configuring the SCA JMS bindings. For complete details, see the topic 'Configuring the SCA JMS binding' in the information center at the address shown here.



Here is a picture detailing the basic steps for configuring JMS binding on a simple SCA application covered in the previous slide.

Composite: Example JMS service binding

```
<?xml version="1.0" encoding="ASCII"?>
<composite xmlns="http://www.osoa.org/xmlns/sca/1.0"
        name="MyValueComposite">

  <service name="MyValueService">
        <interface.java interface="services.myvalue.MyValueService"/>
        <binding.jms>
              <destination name="MyValueServiceQ" create="never"/>
              <activationSpec name="MyValueServiceAS"
  create="never"/>
        </binding.jms>
  </service>
</composite>
```


This example shows the JMS service binding with resources. Note the destination and the activation spec within binding.jms.

[***/binding.jms/activationSpec*** identifies the activation spec that the binding uses to connect to a JMS destination to process request messages. This can be a JNDI name or a plain activation spec name. Note that only JNDI name is supported for the activationSpec.

/binding.jms/response/destination identifies the destination that is to be used to process responses by this binding. It is used to override the destination in the activation specification, and is optional.

Composite: request-response JMS service

```
<composite xmlns="http://www.osoa.org/xmlns/sca/1.0"
    targetNamespace="http://www.ibm.com/soa/sca/samples"
    xmlns:hw="http://www.ibm.com/soa/sca/samples" xmlns:ts="http://tuscany.apache.org/xmlns/sca/1.0"
    name="HelloServiceComposite">
<component name="HelloServiceComponent">
    <implementation.java class="soa.sca.samples.jms.HelloServiceImpl"/>
      <service name="HelloService">
                    <interface.java interface="soa.sca.samples.jms.HelloService"/>
          <binding.jms>
                    <destination name="jms/SCA_sample_Request" type="queue"/>
                    <activationSpec name="jms/SCA_sample_AS"/>
                    <response>
                              <destination name="jms/SCA_sample_Response" type="queue"/>
                              <connectionFactory name="jms/SCA_sample_CF"/>
                    </response>
                              <ts:wireFormat.jmsObject/>
          </binding.jms>
      </service>
 </component>
</composite>
```


This example shows a <binding.jms> element within the component definition file for a request-response message exchange pattern from a JMS client to an SCA service.

The <destination> describes the JMS destination. The destination type is either a queue or topic. This example illustrates the JMS destination queue type. The destination is used to process requests by the JMS binding to the component implementation that contains the service interface

The <activationSpec> element identifies the activation specification that the binding uses to connect to a JMS destination to process request messages. The activation specification name must be a JNDI name. The <activationSpec> element is only supported within the SCA <service> tag.

The <response> element defines the resources used for processing response messages. In this example, the response element specifies the resources for sending messages from the <service> back to the client.

The response <destination> element describes the JMS destination queue that is used to process responses from the service interface.

The response <connectionFactory> element identifies the JNDI name of the connection factory that the binding uses to process response messages.

Example: Two-way JMS SCA reference

```
<reference name="helloWorldService"> <interface.java
  interface="my.HelloWorldService"/>
        <binding.jms>
                    <connectionFactory
  name="jms/helloWorldServiceCF"/>
        <destination name="jms/HelloWorldService"/>
              <response>
                  <destination
  name="jms/SCA_sample_Response"/>
              </response>
        </binding.jms>
</reference>
```


This example describes a <binding.jms> element within the component definition file for a request-response message exchange pattern from a JMS client to an SCA reference.

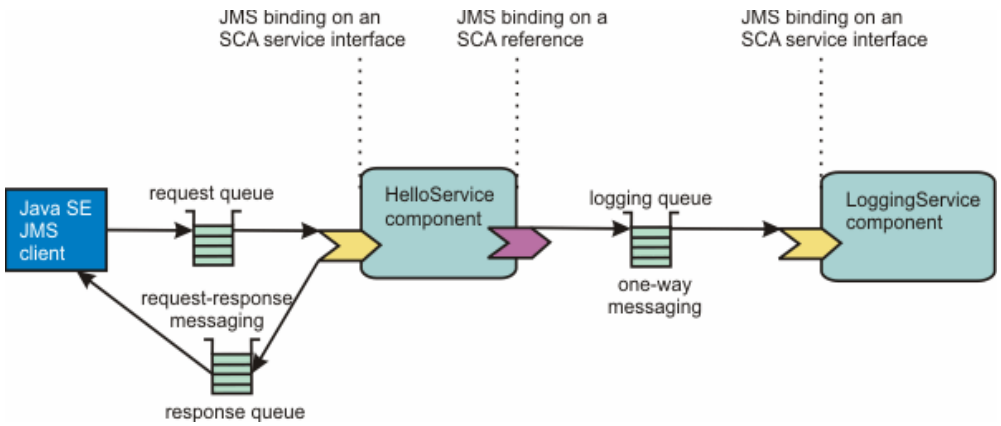
The <connectionFactory> element is used in the SCA reference to identify the JNDI name of the connection factory used to process messages sent from the reference to the referenced service. The <activationSpec> element is not supported in a reference.

The <destination> element is the JMS queue or topic that is used to send messages to the referenced component implementation.

The response <destination> element is the JMS resource that is used to receive response messages to the SCA reference.

Note that the schema for binding.jms requires the destination element to appear before the connectionFactory element.

Example: request-response and one-way



The following example shows two SCA component implementations, HelloService and LoggingService, and the use of request-response and one-way messaging.

The HelloService component implementation illustrates the request-response message pattern. This HelloService component exposes the service interface with the name, getGreeting, that is used to illustrate a return response of hello plus the value of getGreeting.

The LoggingService component implementation is a logging service. This component exposes a one-way service interface with the name log that receives a message and logs the message in a repository.

The HelloService has an SCA reference to the LoggingService. Each time the HelloService service receives a message, it calls the LoggingService service to log the message.

In this example, a Thin Client for JMS application sends a message, formatted as a JMS ObjectMessage message type to the SCA HelloService using the jms/SCA_sample_Request queue. The ObjectMessage sets the scaOperationName property value to getGreetings. The HelloServiceComponent receives the message over the JMS HelloService binding. The HelloServiceComponent then sends a request to the referenced service, LoggingService, and the one-way operation is complete. HelloServiceComponent sends a response of hello plus the value of getGreetings to the client application using the jms/SCA_sample_Response queue to complete the request-response operation.

Section

Summary and references

The next section will provide a summary and references.

Summary

- JMS binding is especially well suited for use by services and references of composites that are directly deployed. This is in comparison to composites that are used as implementations of higher-level components
- For more information on other SCA feature pack supported bindings, reference IBM Education assistant SCA materials published in version 1.0

The binding is especially well suited for use by services and references of composites. These services and references are directly deployed. Composites are used as implementations of higher-level components. Services and references of deployed composites become system-level services and references, which are intended to be used by non-SCA clients.

The messaging binding describes a common pattern of behavior that can be followed by messaging-related bindings, including the JMS binding. In particular it describes the manner in which operations are selected based on message content, and the manner in which messages are mapped into the runtime representation. These are specified in a language-neutral manner. The JMS binding provides JMS-specific details of the connection to the required JMS resources. It supports the use of Queue and Topic type destinations.

For more information on other SCA feature pack supported bindings, reference the IBM Education Assistant SCA 1.0 presentations.

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

- 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

References

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_Overview.ppt

This module is also available in PDF format at: ../WASv7SCA101_JMSbinding_Overview.pdf

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:

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.