IBM Software Group

IBM® WebSphere® Application Server V7 Feature Pack for Service Component Architecture

Quality of service - Transactions

@business on demand.

Updated December 10, 2008

This presentation covers Global and local transactions of the Quality of service for the SCA feature pack.

Global transactions (1)

- Components that use a synchronous interaction style can be part of a single, distributed ACID transaction
- **managedTransaction.global**

```
<component name="DataAccessComponent">
    <implementation.java class="example.DataAccessImpl"
         requires="managedTransaction.global"/>
</component>
```

- **propagatesTransaction** or **suspendsTransaction** controls whether a component's service runs under its client's global

The SCA transaction specification describes the service component's transactional environment as managedTrasaction and "managed transactions" are described in terms of either "global" or "local" transactions. The "managed" aspect of managed transactions refers to the transaction environment provided by the SCA runtime for the business component, which can interact with other business components and with resource managers. The managed transaction environment defines the transactional context under which such interactions occur.

From an SCA perspective, a global transaction is a unit-of-work scope within which transactional work is atomic. For example, if multiple transactional resource managers are accessed under a global transaction, then the transactional work is coordinated to either atomically commit or rollback regardless using a "two phase commit" protocol. A global transaction can be propagated on synchronous invocations between components such that multiple, remote service providers can run distributed requests under the same global transaction.

Components that use a synchronous interaction style can be part of a single, distributed ACID (atomic, consistent, isolated, and durable) transaction, within which all transaction resources are coordinated to either atomically commit or rollback.

This is specified using the **managedTransaction.global** intent in the requires attribute of the <implementation.java> element as shown here. You can control whether a component's service runs under its client's global transaction by specifying either the **propagatesTransaction** or **suspendsTransaction** intent on the component's <service> element.

Global transactions (2)

- **propagatesTransaction** – The service runs under its client's global transaction. **suspendsTransaction** – The service runs in its own global transaction separate from the client's transaction
- Transaction context is never propagated on @OneWay methods

propagatesTransaction refers to the service running under its client's global transaction. If the client is not running in a global transaction or chose not to propagate its global transaction, the service runs in its own global transaction.

suspendsTransaction refers to the service running in its own global transaction separate from the client's transaction.

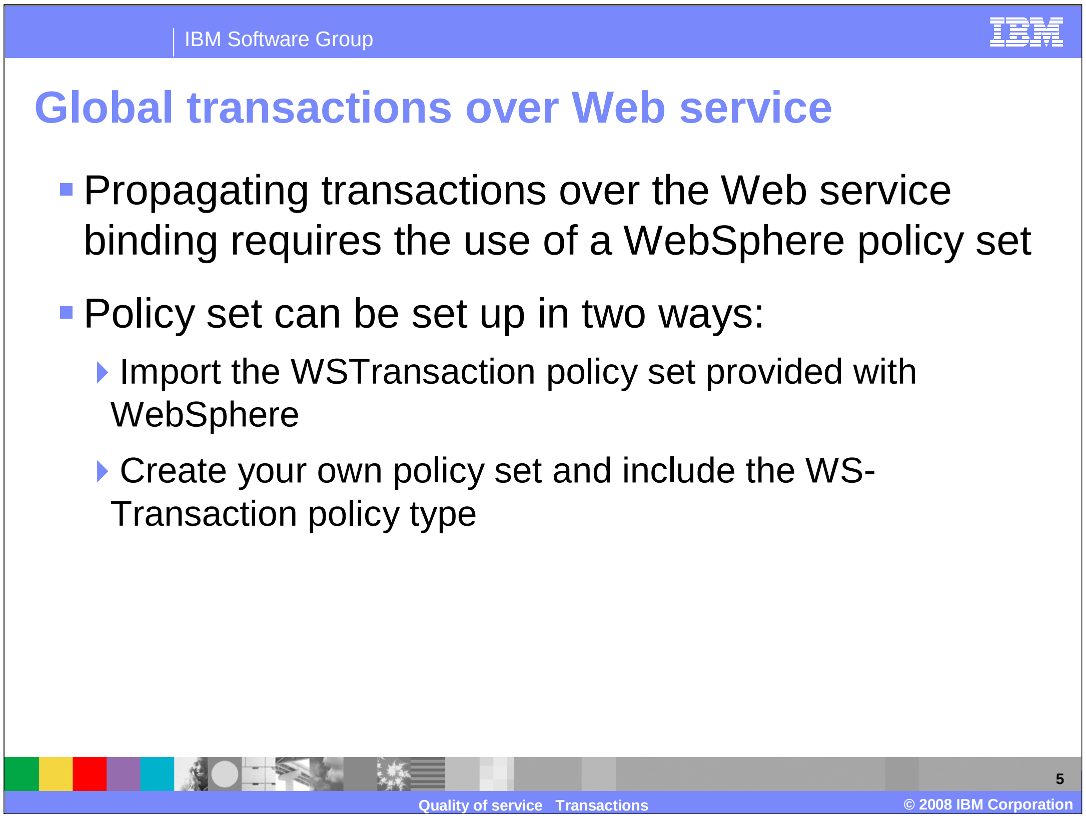
Transaction context is never propagated on @OneWay methods. The SCA runtime ignores propagatesTransaction for OneWay methods.

Global transactions : Intents

- Use of the managedTransaction.global, suspendsTransaction and propagatesTransaction intents

```
<component name="DataUpdateComponent">
    <implementation.java class="example.DataUpdateImpl"
            requires="managedTransaction.global"/>
    <service name="DataUpdateService"
            requires="suspendsTransaction"/>
    <reference name="myDataAccess" target="DataAccessComponent"
            requires="propagatesTransaction"/>
</component>
```


This example shows the use of the managedTransaction.global, suspendsTransaction and propagatesTransaction intents. The DataUpdateComponent runs in its own global transaction, not in its client's transaction, because suspendsTransaction is specified on its <service> element. Its global transaction is propagated to the referenced service DataAccessComponent because propagatesTransaction is specified on its <reference> element



Propagating transactions over the Web service binding requires the use of a WebSphere policy set that contains the WS-Transaction policy type. You can set up this policy set in one of two ways:

You can import the WSTransaction policy set provided with WebSphere, or you can create your own policy set and include the WSTransaction policy type.

Global transactions: Composite

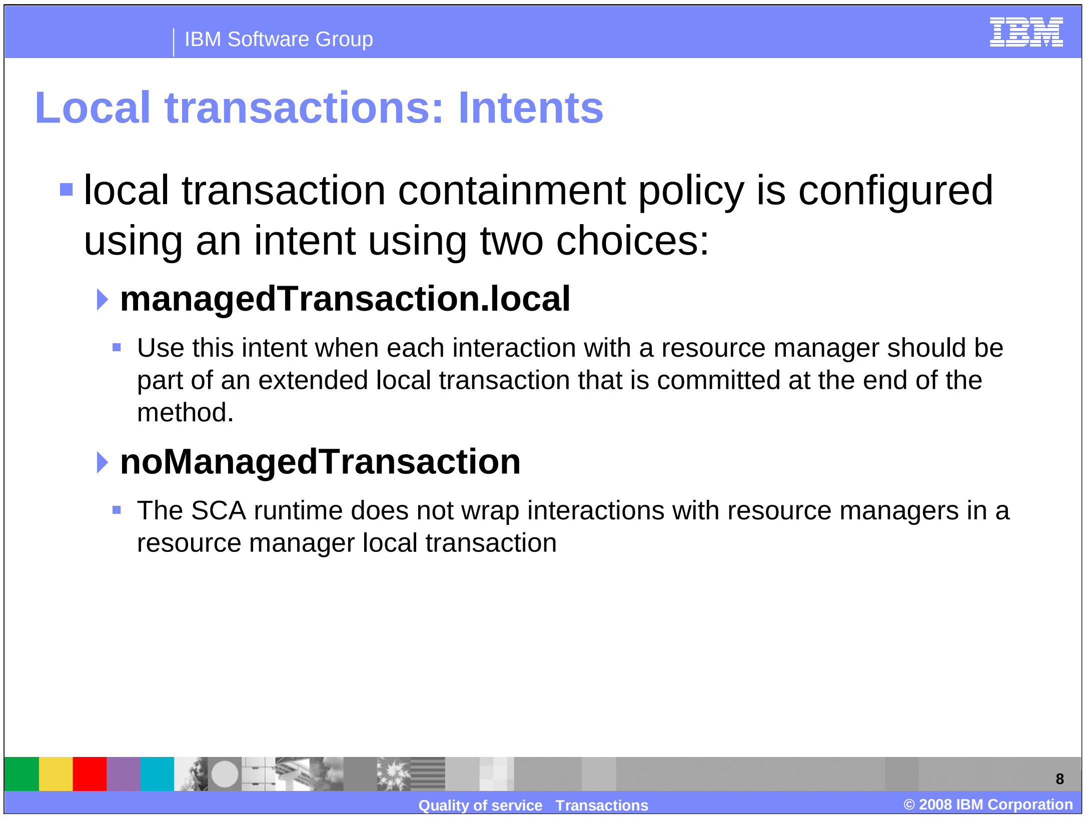
```
composite name="WSDataUpdateComposite"
 xmlns="http://www.osoa.org/xmlns/sca/1.0"
 xmlns:ws="http://www.ibm.com/xmlns/prod/websphere/sca/1.0/2007/06">
    <component name="WSDataUpdateComponent">
       <implementation.java class="example.DataUpdateImpl"
            requires="managedTransaction.global"/>
       <service name="DataUpdateService"
            requires="propagatesTransaction">
            <binding.ws ws:wsPolicySet="WSTransaction"/>
       </service>
       <reference name="myDataBuddy" target="DataBuddyComponent"
            requires="propagatesTransaction">
            <binding.ws ws:wsPolicySet="WSTransaction"/>
       </reference>
    </component>
</composite>
```


Here is an example that shows the use of the WSTransaction policy set. Remember you can create your own policy set and include the WS-Transaction policy type.

Local transactions

- A component can be configured to run under local transaction containment (LTC)
- The component's interactions with resource providers (such as databases) are managed within resource manager local transactions (RMLTs)
- Resource manager local transaction (RMLT) represents a unit of recovery on a single connection that is managed by the resource manager

From an SCA perspective, Business logic might need to access transactional resource managers without the presence of a global transaction. A component can be configured to run under local transaction containment (LTC). The SCA runtime starts an LTC before dispatching a method on the component and completes the LTC at the end of the method dispatch. The component's interactions with resource providers (such as databases) are managed within resource manager local transactions. A resource manager local transaction represents a unit of recovery on a single connection that is managed by the resource manager.



The local transaction containment policy is configured using an intent. There are two choices:

Use **managedTransaction.local** when each interaction with a resource manager should be part of an extended local transaction that is committed at the end of the method. The SCA runtime wraps interactions with each resource manager in a resource manager local transaction, or RMLT. The SCA runtime commits each RMLT at the end of method dispatch, unless an unchecked exception occurs, in which case the SCA runtime aborts each RMLT. The component may not use resource manager commit/rollback interfaces or set AutoCommit to true. If multiple resource managers are used, the RMLTs are committed independently so it is possible for some to fail and some to succeed. If this is undesireable, use a global transaction.

If you use **noManagedTransaction**, the SCA runtime does not wrap interactions with resource managers in a RMLT. The component implementation manages the start and end of its own RMLTs or gets AutoCommit behavior (which commits after each use of a resource) by default. The component must complete any RMLTs before the end of the method dispatch otherwise the SCA runtime will stop them.

Local transactions: Attribute

- Local Transaction intent is specified using the requires attribute on the <implementation.java> element

```
<component name="DataAccessLocalComponent">
    <implementation.java class="example.DataAccessImpl"
              requires="managedTransaction.local"/>
</component>
```

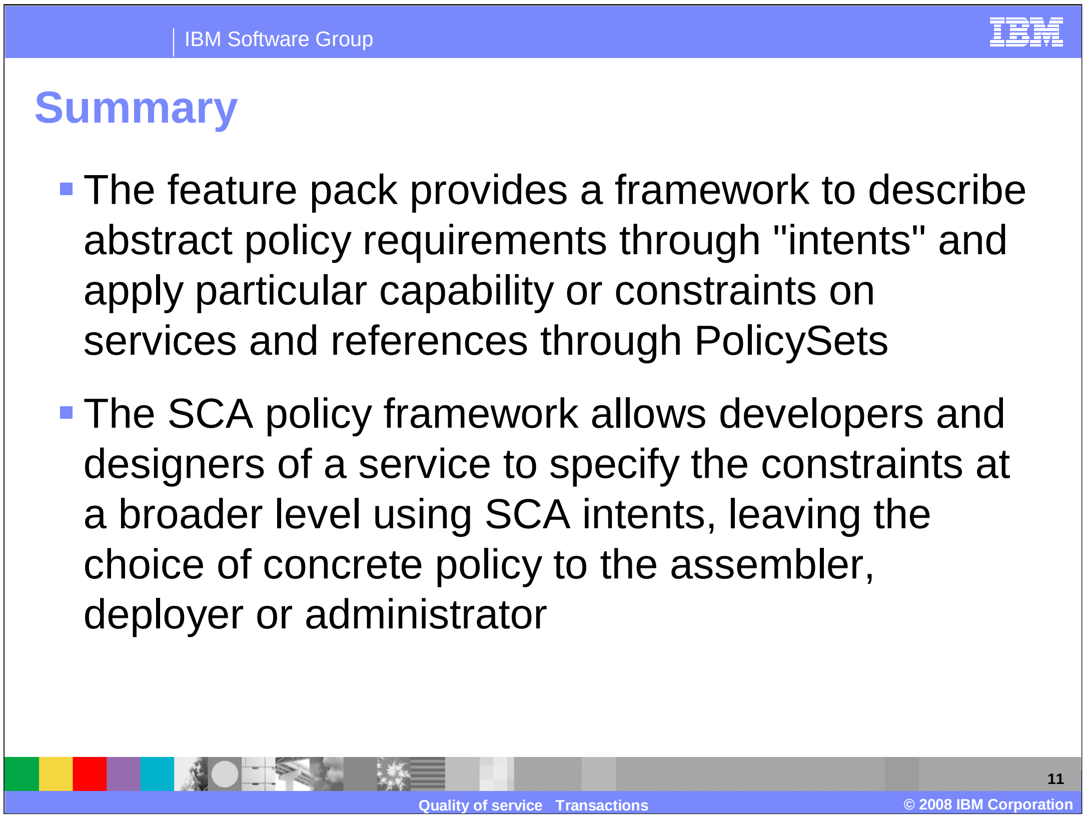
- A local transaction cannot be propagated from one component to another

This example shows how the local transaction intent is specified using the requires attribute on the <implementation.java> element. A local transaction cannot be propagated from one component to another. It is an error to specify propagatesTransaction on a component’s <service> if the component uses the managedTransaction.local or noManagedTransaction intent.

Transactions intents: Limitations

- If transactional intents are not specified then the default behavior is vendor-specific
- In the SCA feature pack, if a transactional intent in not specified for the implementation, the default is managedTransaction.global
- If a transactional intent is not specified for a service or reference, the default is suspendsTransaction

If transactional intents are not specified then the default behavior is vendor-specific. In the SOA feature pack, if a transactional intent is not specified for the implementation, the default is managedTransaction.global. If a transactional intent is not specified for a service or reference, the default is suspendsTransaction. You should specify the required intents rather than rely on default behavior so that the application is portable. In the appendix at the end of this presentation is a table showing intents supported by each binding.



In summary, the feature pack provides a framework to describe abstract policy requirements through "intents" and apply particular capability or constraints on services and references through PolicySets.

The SCA policy framework allows developers and designers of a service to specify the constraints at a broader level using SCA intents, leaving the choice of concrete policy to the assembler, deployer or administrator. This empowers the assembler to provide a combination of services that can behave differently based on its operating environment and need, without changing the underlying business logic itself, keeping with the SCA concepts.

The next several slides are an appendix that contains list of supported intents and a list of old and new names for intents.

Section

Appendix

Intents supported by each binding

Intent	binding.ws	binding.ejb binding.sca
authentication.message	Requires the attachment of a WebSphere policy set and policy binding that contains the WS-Security policy type	Not supported; CSIv2 can be configured to use basic auth and security token (LTPA, Kerberos)
confidentiality.message integrity.message	Requires the attachment of a WebSphere policy set and policy binding that contains the Security policy type	Not supported

Intents supported by each binding.

Intents supported by each binding

Intent	binding.ws	binding.ejb binding.sca
authentication.transport	Basic auth only. Reference requires the attachment of a WebSphere policy set that contains the HTTPTransport policy type. Service does not require any attachments.	Intent is not supported. CSIv2 can be configured to use client certificates for authentication.
confidentiality.transport integrity.transport	Requires the attachment of a WebSphere policy set that contains the SSLTransport policy type	Intent is not supported. CSIv2 can be configured to require SSL.

Intents supported by each binding

Intent	binding.ws	binding.ejb binding.sca
propagatesTransaction	Requires the attachment of a Web services policy set that contains the WS-Transaction policy type	Supported; no configuration required

Old and new intent names

SOA feature pack for WebSphere 6.1	SCA feature pack for WebSphere 7.0
managedTransaction.global (default)	managedTransaction.global (default)
managedTransaction.local	managedTransaction.local
managedTransaction.none	noManagedTransaction
managedTransaction.any	No equivalent intent
propagatesTransaction.true (default)	propagatesTransaction
propagatesTransaction.false	suspendsTransaction (default)

The SOA feature pack for WebSphere 6.1 used different names for some of the transactional intents and different default intents. This table shows the old and new intent names and the default intents.

References

- SCA specifications

http://www.osoa.org/display/Main/Service+Component+Architecture+Specifications

- SCA policy framework

http://www.osoa.org/download/attachments/35/SCA_Policy_Framework_V100.pdf?version=1

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_WASv7SCA_QOS_GlobalTransactions.ppt

This module is also available in PDF format at: ../WASv7SCA_QOS_GlobalTransactions.pdf

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

Trademarks, copyrights, and disclaimers