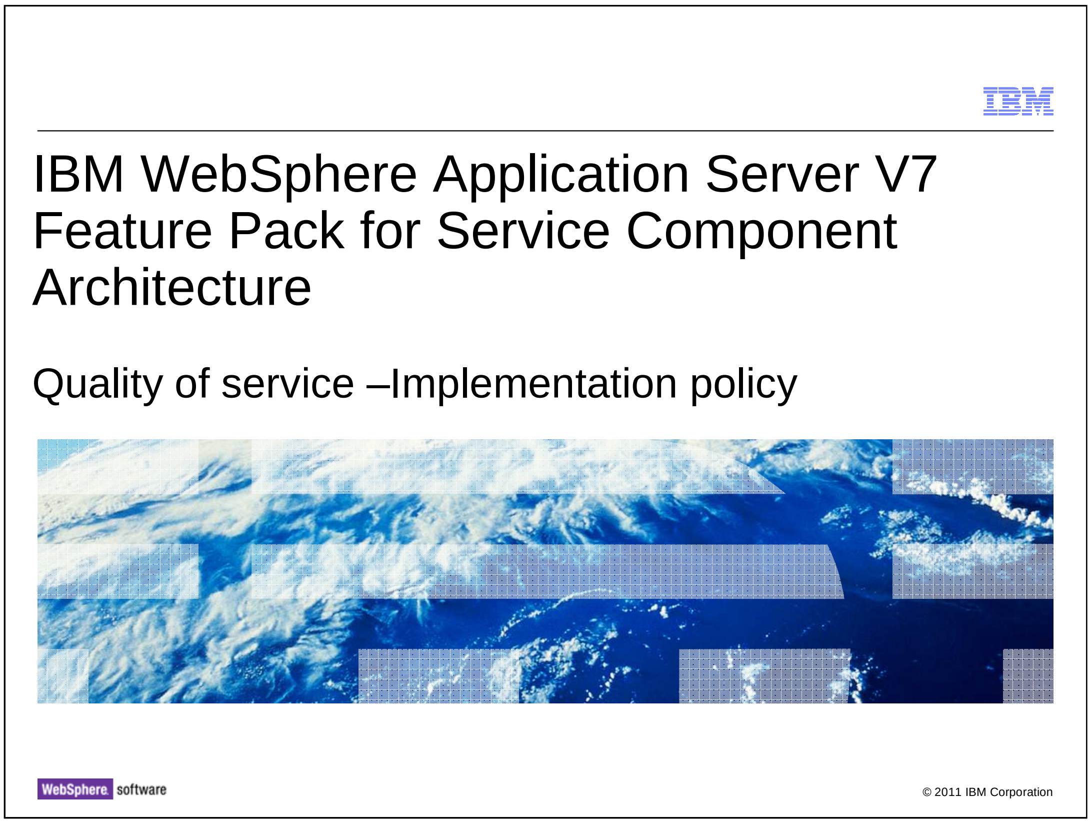
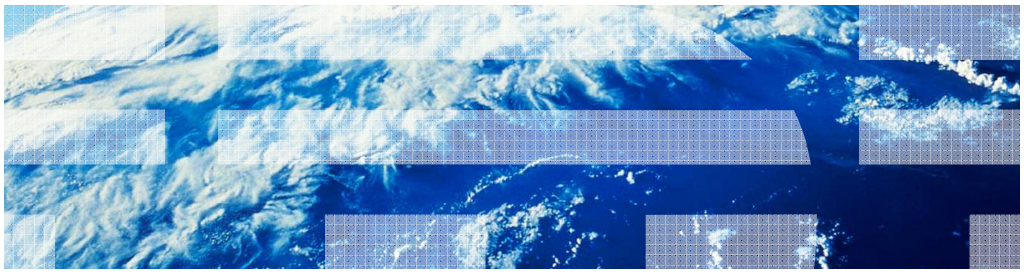


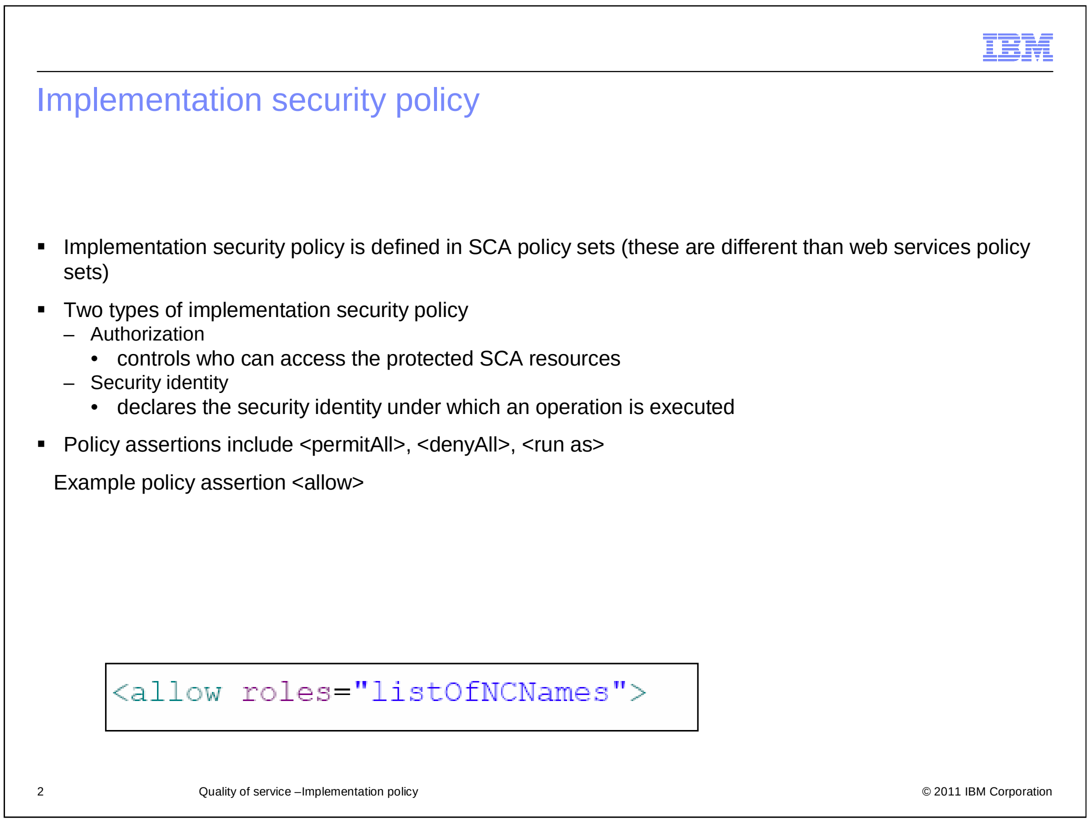
IBM WebSphere Application Server V7 Feature Pack for Service Component Architecture

Quality of service –Implementation policy



WebSphere software

This presentation covers the implementation policy of a service for the SCA feature pack.



Implementation security policy

- Implementation security policy is defined in SCA policy sets (these are different than web services policy sets)
- Two types of implementation security policy
  - Authorization
    - controls who can access the protected SCA resources
  - Security identity
    - declares the security identity under which an operation is executed
- Policy assertions include <permitAll>, <denyAll>, <run as>

Example policy assertion <allow>

```
<allow roles="listOfNCNames">
```

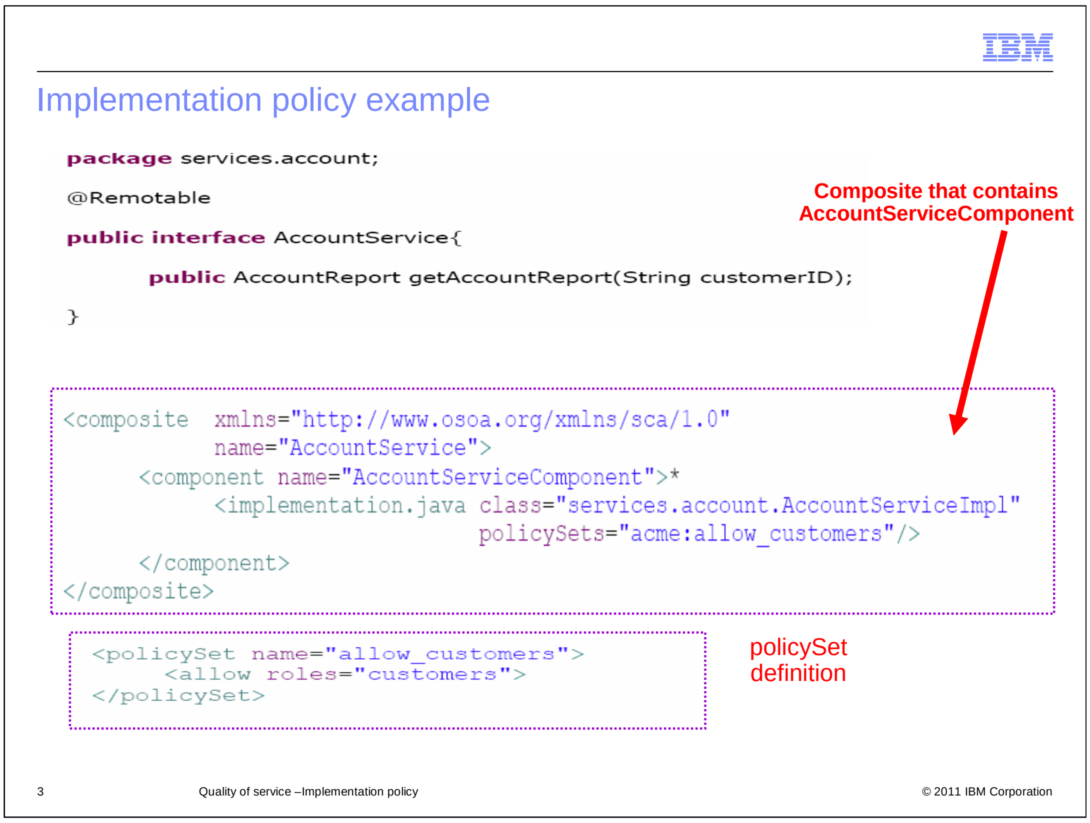
2 Quality of service –Implementation policy © 2011 IBM Corporation

Two policy assertions are defined which apply to implementations – ***Authorization*** and ***Security Identity***.

Authorization controls who can access the protected SCA resources. A security role is an abstract concept that represents a set of access control constraints on SCA resources such as composites, components, and operations. The approach and scope of the mapping of role names to security principals is SCA runtime implementation dependent. Scope implies the set of artifacts contained by some higher-level artifact, so that a composite contains components, a component contains services and references, services and references contain an interface, and an interface contains operations.

Security Identity declares the security identity under which an operation is executed. Both are represented as policy assertions that are used within policySets created for implementations (like implementation policies).

In the example shown here, the policy assertion is “allow”. When the <allow> element is included in a policySet used on a component, that component can only be accessed by principal. The principal role corresponds to one of the role names listed in the @roles attribute. How role names are mapped to security principals is implementation dependent, because SCA does not define this.



Implementation policy example

```
package services.account;
@Remotable
public interface AccountService{
        public AccountReport getAccountReport(String customerID);
}
```

Composite that contains AccountServiceComponent

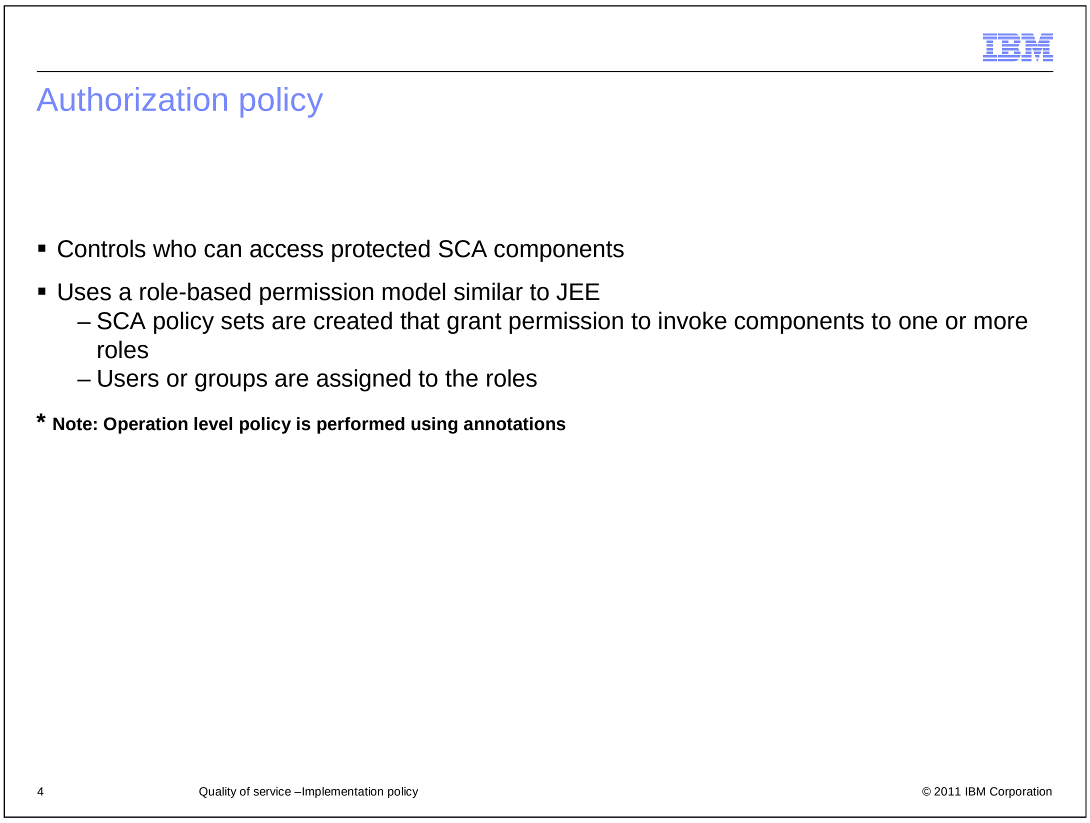
```
<composite  xmlns="http://www.osoa.org/xmlns/sca/1.0"
            name="AccountService">
      <component name="AccountServiceComponent">*
            <implementation.java class="services.account.AccountServiceImpl"
                                 policySets="acme:allow_customers"/>
      </component>
</composite>
```

```
<policySet name="allow_customers">
     <allow roles="customers">
</policySet>
```

policySet definition

3 Quality of service –Implementation policy © 2011 IBM Corporation

Shown here is an example implementation, written in Java, The AccountService implements the ***AccountService*** interface, which is defined through a Java interface. Notice the policySets and policySet definition. The composite that contains an AccountServiceComponent should be accessible by anyone with the “customer” role.



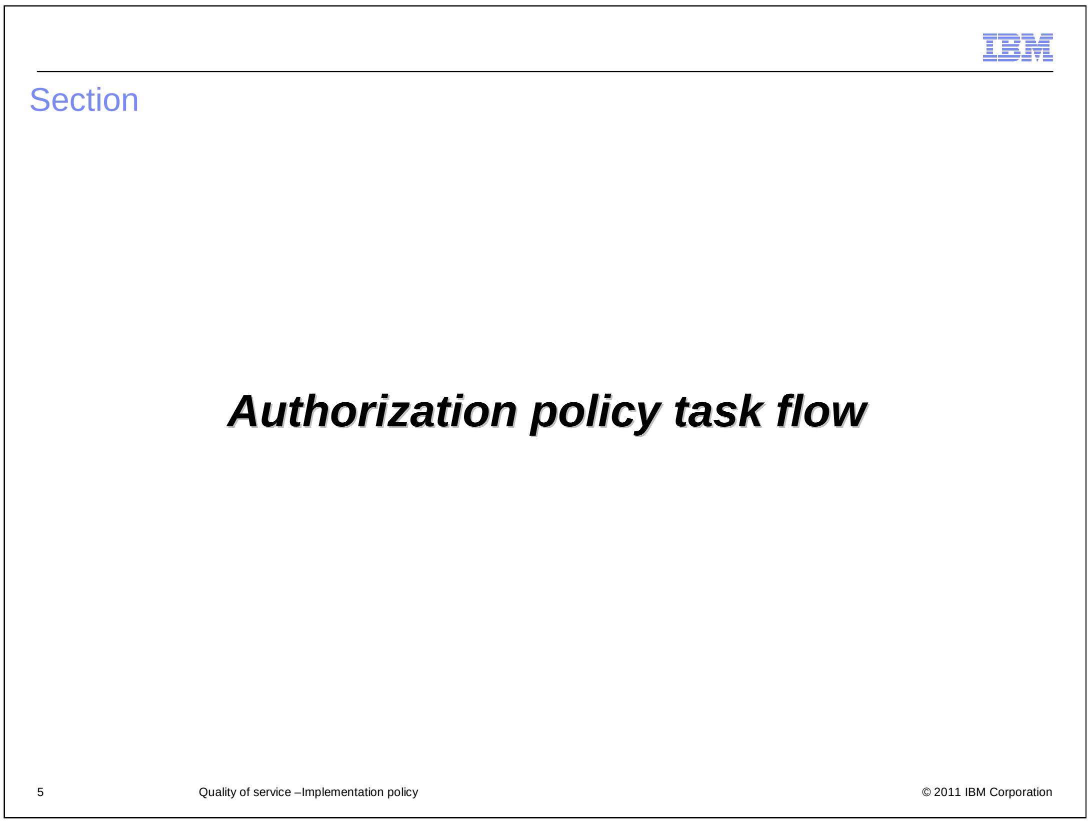
Authorization policy

- Controls who can access protected SCA components
- Uses a role-based permission model similar to JEE
  - SCA policy sets are created that grant permission to invoke components to one or more roles
  - Users or groups are assigned to the roles

*** Note: Operation level policy is performed using annotations**

Authorization policy controls who can access protected SCA components and operations and applies to implementations. Authorization controls who can access the protected SCA resources. A security role is an abstract concept that represents a set of access control constraints on SCA resources such as composites, components, and operations. The approach and scope of the mapping of role names to security principals is SCA runtime implementation dependent. Security Identity policy declares the security identity under which an SCA component or operation is executed, allowing you to limit access to an SCA component or operation to particular users or groups, and to delegate to another user when executing an SCA component or operation.

Note: Operation level policy is performed using annotations. Policy set attachments to operations are not permitted in the composite file.

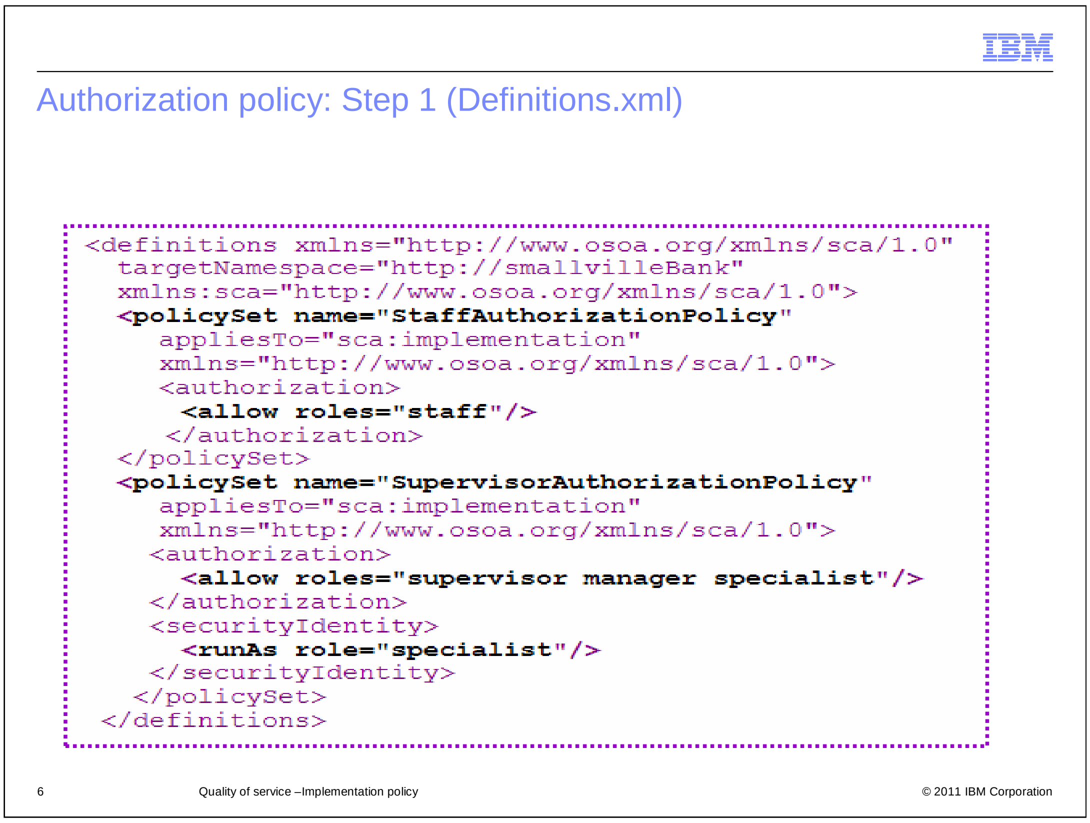


Section

Authorization policy task flow

5 Quality of service –Implementation policy © 2011 IBM Corporation

This section describes the steps necessary to set up authorization policy



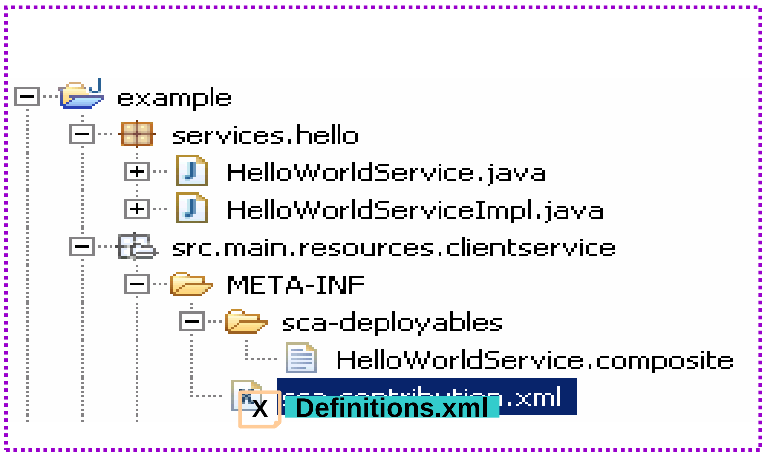
Authorization policy: Step 1 (Definitions.xml)

```
<definitions xmlns="http://www.osoa.org/xmlns/sca/1.0"
  targetNamespace="http://smallvilleBank"
  xmlns:sca="http://www.osoa.org/xmlns/sca/1.0">
  <policySet name="StaffAuthorizationPolicy"
     appliesTo="sca:implementation"
     xmlns="http://www.osoa.org/xmlns/sca/1.0">
     <authorization>
       <allow roles="staff"/>
      </authorization>
  </policySet>
  <policySet name="SupervisorAuthorizationPolicy"
     appliesTo="sca:implementation"
     xmlns="http://www.osoa.org/xmlns/sca/1.0">
    <authorization>
       <allow roles="supervisor manager specialist"/>
    </authorization>
    <securityIdentity>
       <runAs role="specialist"/>
    </securityIdentity>
   </policySet>
 </definitions>
```

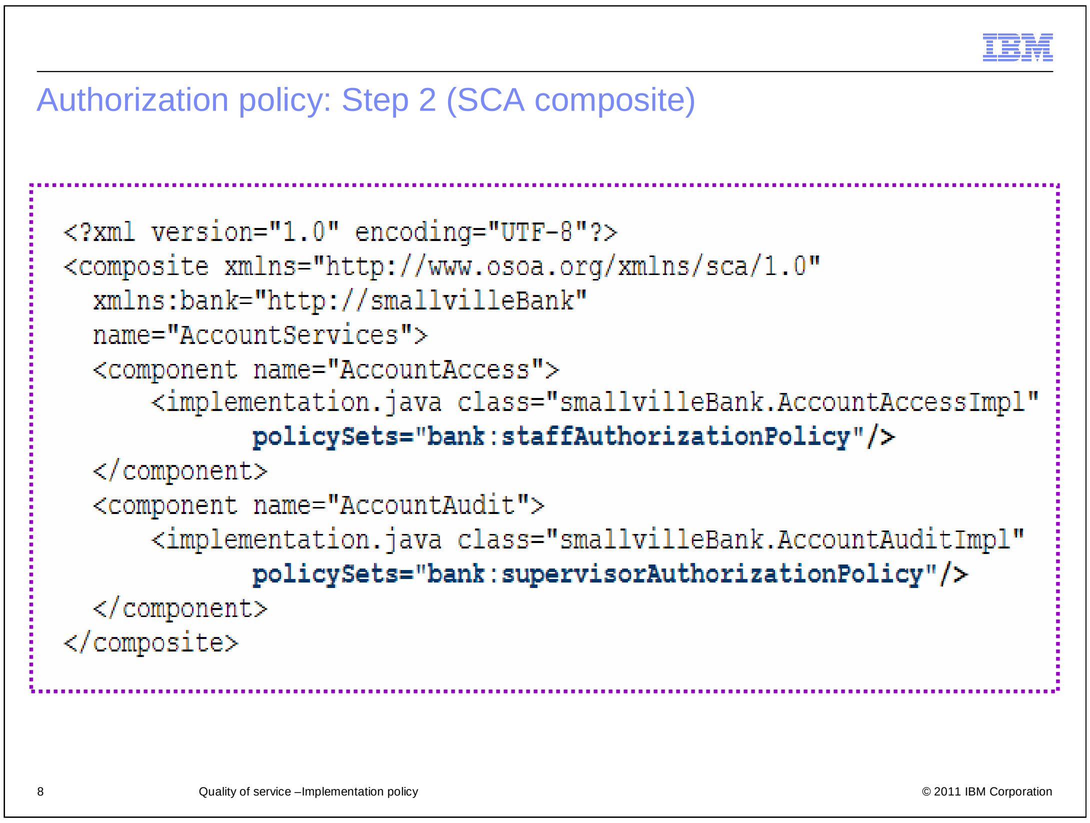
Step 1:

First, the policy administrator creates one or more policy sets in the file named definitions.xml as shown in this example. This file is normally packaged at the root of META-INF directory of the jar. Notice the policySet names, user roles and runAs roles.

Definitions.xml



This graphic shows you exactly where the Definitions.xml file is located in your SCA jar file. The definitions.xml file can be at any location in the jar; it does not have to be under META-INF.

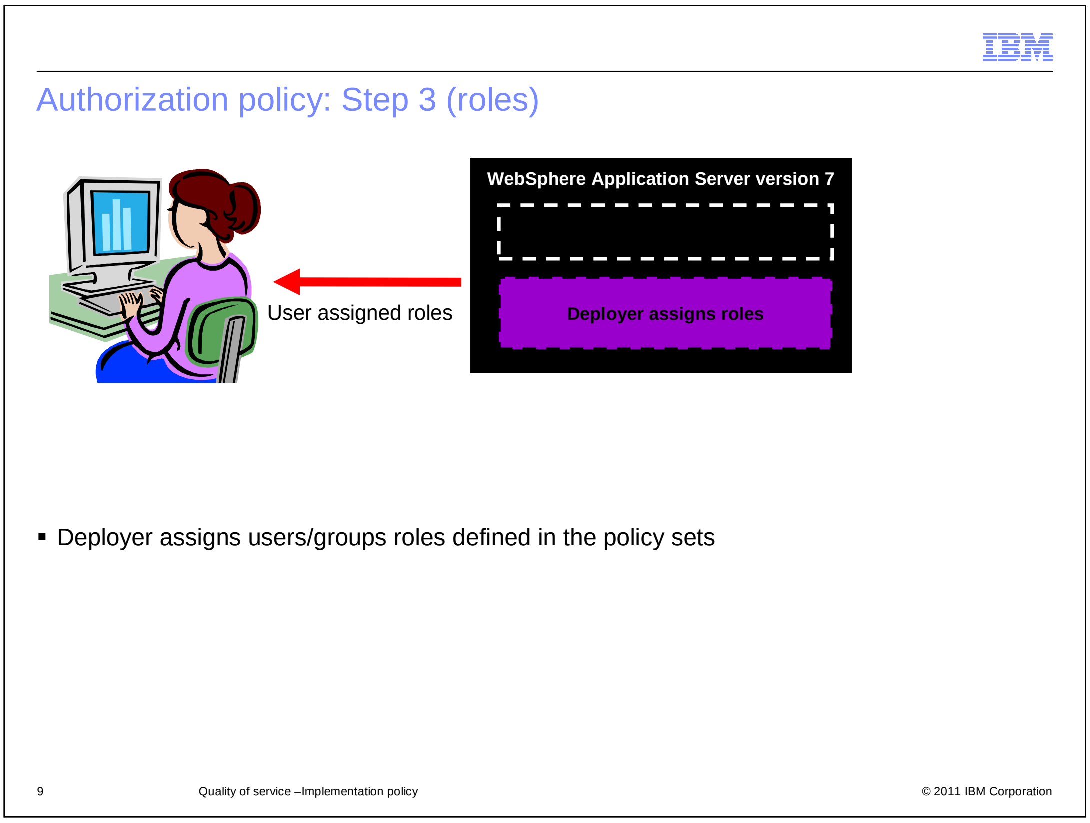


Authorization policy: Step 2 (SCA composite)

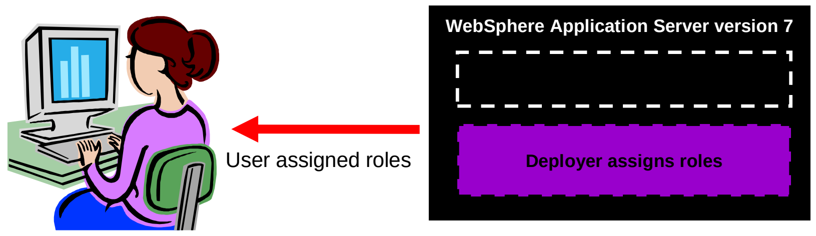
```
<?xml version="1.0" encoding="UTF-8"?>
<composite xmlns="http://www.osoa.org/xmlns/sca/1.0"
  xmlns:bank="http://smallvilleBank"
  name="AccountServices">
  <component name="AccountAccess">
      <implementation.java class="smallvilleBank.AccountAccessImpl"
             policySets="bank:staffAuthorizationPolicy"/>
  </component>
  <component name="AccountAudit">
      <implementation.java class="smallvilleBank.AccountAuditImpl"
             policySets="bank:supervisorAuthorizationPolicy"/>
  </component>
</composite>
```

Step 2:

Next, the assembler attaches the policy set to the SCA composite as in the example shown to protect each component. Attaching no policy set will result in access to all users. The policySets names specified are the same names that you saw in the definitions.xml file.



Authorization policy: Step 3 (roles)



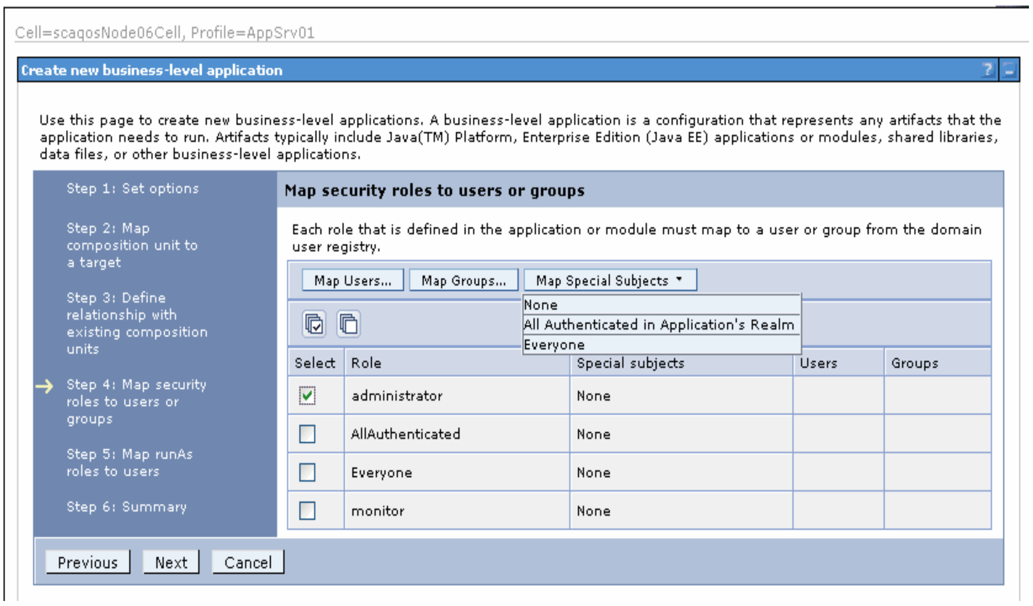
- Deployer assigns users/groups roles defined in the policy sets

9 Quality of service –Implementation policy © 2011 IBM Corporation

Step 3:

Next, the deployer assigns users and or groups to the roles that are defined in the policy sets referenced in the composite file. This is done when creating a new business level application and can be done through scripting or using the administrative console. This is very similar to the role mapping that is done for JEE applications.

Map security roles to users


Cell=scaqosNode06Cell, Profile=AppSrv01

Create new business-level application

Use this page to create new business-level applications. A business-level application is a configuration that represents any artifacts that the application needs to run. Artifacts typically include Java(TM) Platform, Enterprise Edition (Java EE) applications or modules, shared libraries, data files, or other business-level applications.

Step 1: Set options
Step 2: Map composition unit to a target
Step 3: Define relationship with existing composition units
→ Step 4: Map security roles to users or groups
Step 5: Map runAs roles to users
Step 6: Summary

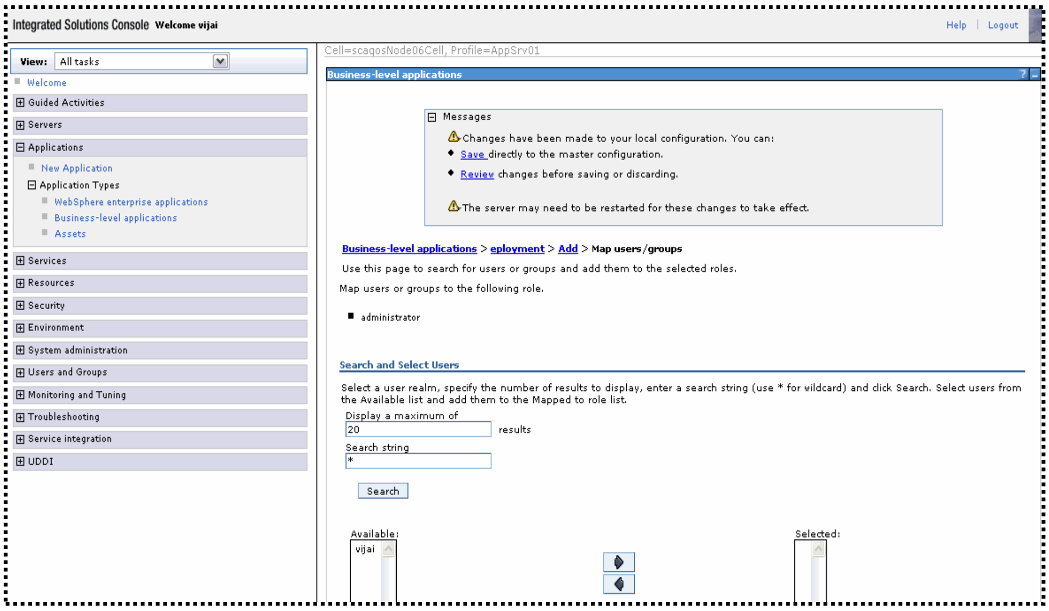
Map security roles to users or groups

Each role that is defined in the application or module must map to a user or group from the domain user registry.

Select	Role	Special subjects	Users	Groups
✓	administrator	None		
	AllAuthenticated	None		
	Everyone	None		
	monitor	None		

This is the screen you use to work with roles through the administrative console when creating a business level application.

Map users to the roles


Integrated Solutions Console Welcome vijai

Cell=scaqosNode06Cell, Profile=AppSrv01

Business-level applications

Messages

Changes have been made to your local configuration. You can:

- Save directly to the master configuration.
- Review changes before saving or discarding.

The server may need to be restarted for these changes to take effect.

Business-level applications > eployment > Add > Map users/groups

Use this page to search for users or groups and add them to the selected roles.

Map users or groups to the following role.

- administrator

Search and Select Users

Select a user realm, specify the number of results to display, enter a search string (use * for wildcard) and click Search. Select users from the Available list and add them to the Mapped to role list.

Display a maximum of 20 results

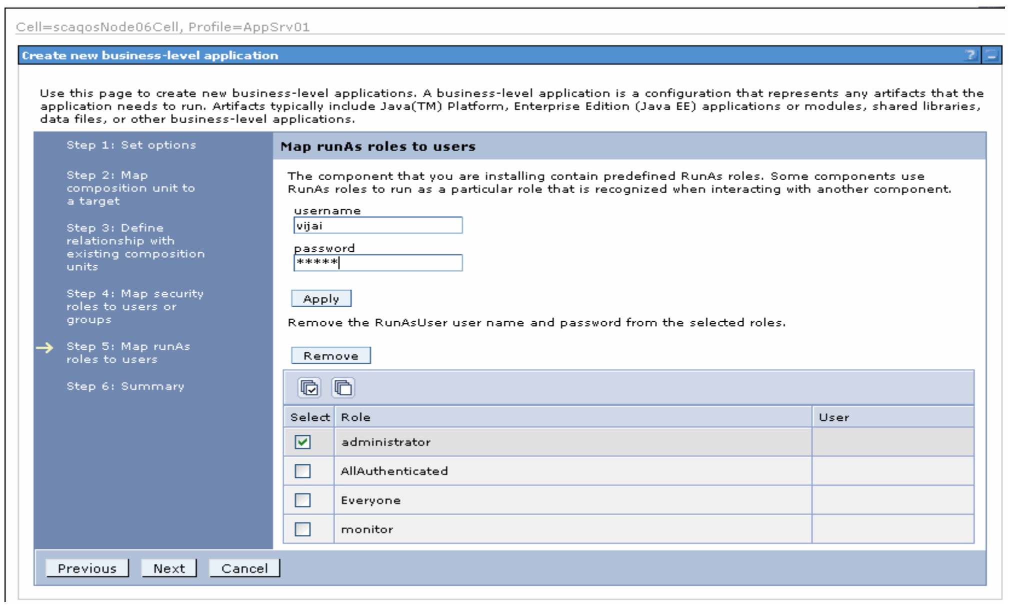
Search string *

Available: vijai

Selected:

The next step is to search for users and groups to add them to the selected roles.

Authorization policy: Step 4 (runAs roles)


Cell=scaqosNode06Cell, Profile=AppSrv01

Create new business-level application

Use this page to create new business-level applications. A business-level application is a configuration that represents any artifacts that the application needs to run. Artifacts typically include Java(TM) Platform, Enterprise Edition (Java EE) applications or modules, shared libraries, data files, or other business-level applications.

Step 1: Set options
Step 2: Map composition unit to a target
Step 3: Define relationship with existing composition units
Step 4: Map security roles to users or groups
Step 5: Map runAs roles to users
Step 6: Summary

Map runAs roles to users

The component that you are installing contain predefined RunAs roles. Some components use RunAs roles to run as a particular role that is recognized when interacting with another component.

username: vijai

password: *****

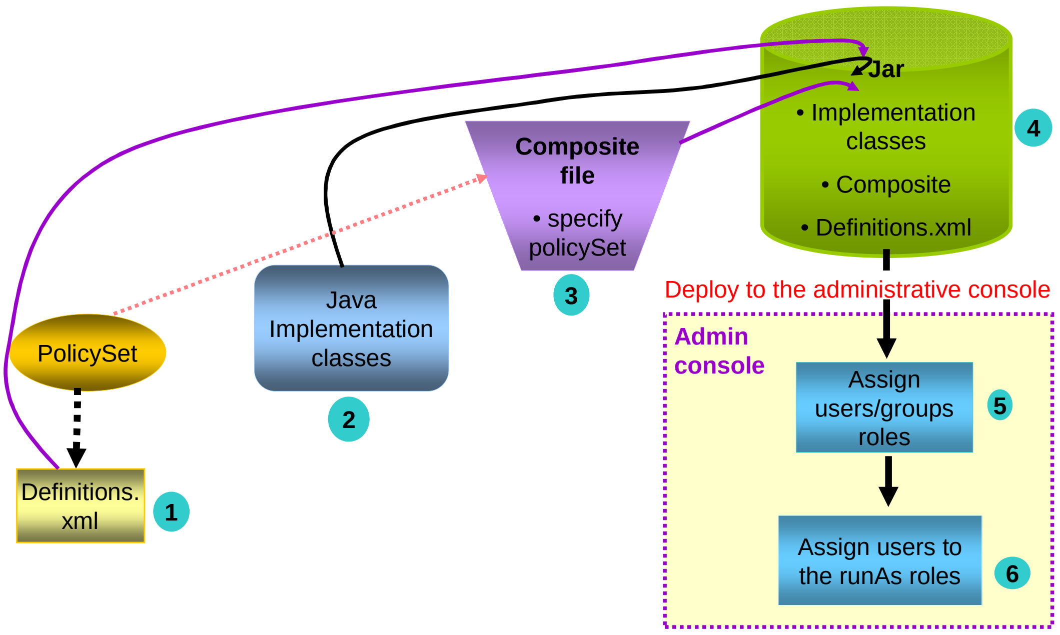
Remove the RunAsUser user name and password from the selected roles.

Select	Role	User
✓	administrator	
☐	AllAuthenticated	
☐	Everyone	
☐	monitor	

Step 4:

Next, the deployer assigns a runAs role to users as defined in the policy sets referenced in the composite file.

Authorization policy summary



In summary, here are the steps necessary to set up an authorization policy.

1) Create a PolicySet - defined in the definitions.xml.

2) Create implementation classes and compile - Policy can be annotated also

3) Create a composite file specifying the policySet to declare the SCA composite

4) Create a jar file containing the implementation classes, the composite file and possibly the contribution file and definitions.xml.

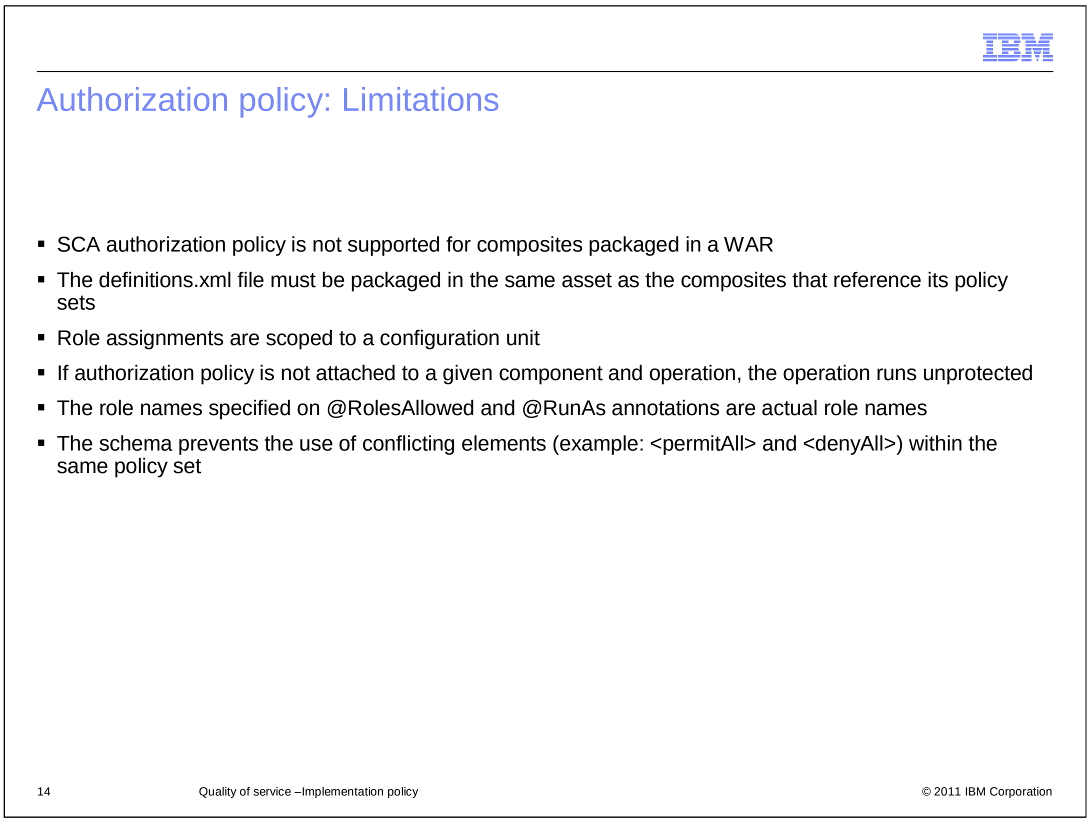
5) Deploy the jar file into the administrative console as an asset that can be added to a business level application.

> 5b) During deployment of the jar assign users/groups to the roles defined in the composite file.

6) Assign users to the runAs role defined in the composite.

7) You are now ready to start your application

Administrative and application security must be enabled at the beginning before deploying the application.



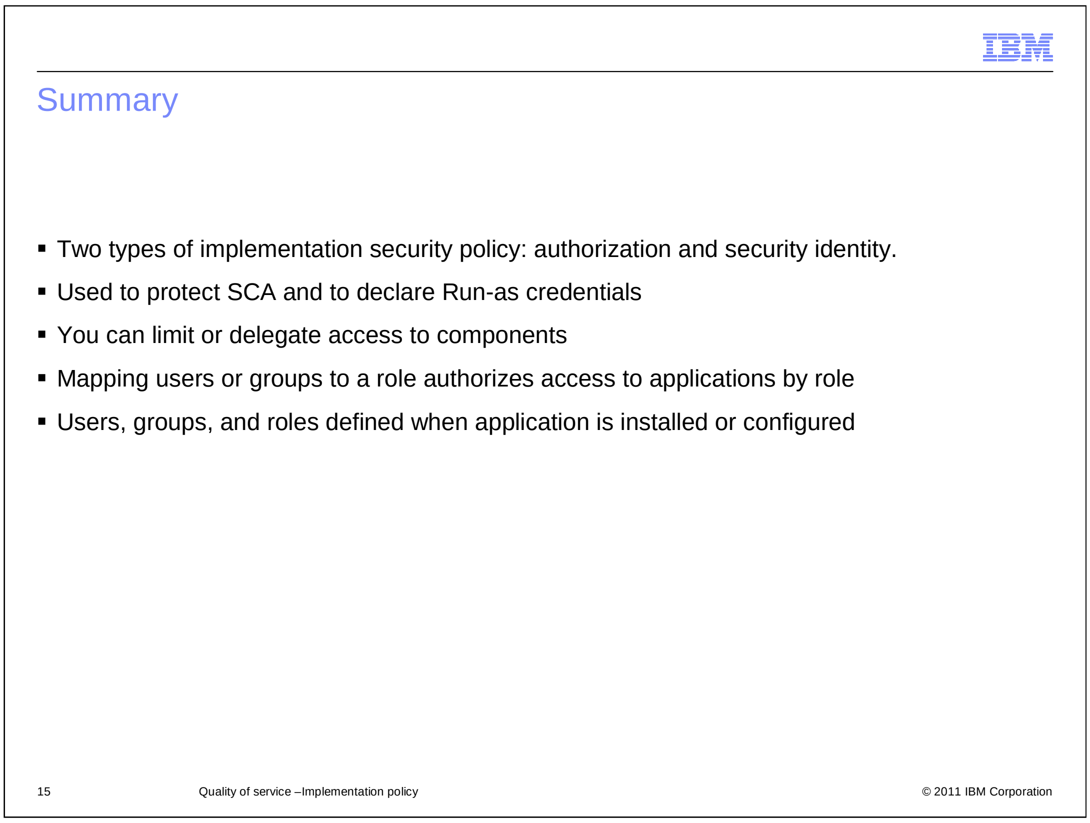
Authorization policy: Limitations

- SCA authorization policy is not supported for composites packaged in a WAR
- The definitions.xml file must be packaged in the same asset as the composites that reference its policy sets
- Role assignments are scoped to a configuration unit
- If authorization policy is not attached to a given component and operation, the operation runs unprotected
- The role names specified on @RolesAllowed and @RunAs annotations are actual role names
- The schema prevents the use of conflicting elements (example: <permitAll> and <denyAll>) within the same policy set

SCA authorization policy is only supported for composites packaged in a .WAR file. The definitions.xml file must be packaged in the same asset as the composites that reference its policy sets. That is, it is not supported to package the policy sets separately from the composites. Role assignments are scoped to a configuration unit. Role assignments are required for all roles used in all composites within the configuration unit. These role assignments are completely independent of any role assignments made for other configuration units in the same business level application. If an authorization policy is not attached to a given component and operation, the operation runs unprotected. The authorization policy does not have the option to run as system identity. The role names specified on @RolesAllowed and @RunAs annotations are actual role names. There is no "role reference" capability to map roles used within code to actual roles as there is in EJB. The schema prevents the use of conflicting elements (example. <permitAll> and <denyAll>) within the same policy set. However it is possible to create conflicts by specifying multiple policy sets in the @policySets attribute or by inheriting policy sets across elements. In this case, these rules are used:

> <denyAll> takes precedence over <permitAll>, which takes precedence over <allow>.
>
> Roles from multiple <allow> elements and from @RolesAllowed annotations are aggregated.

Note: Operation level policy is performed using annotations. Policy set attachments to operations are not permitted in the composite file.

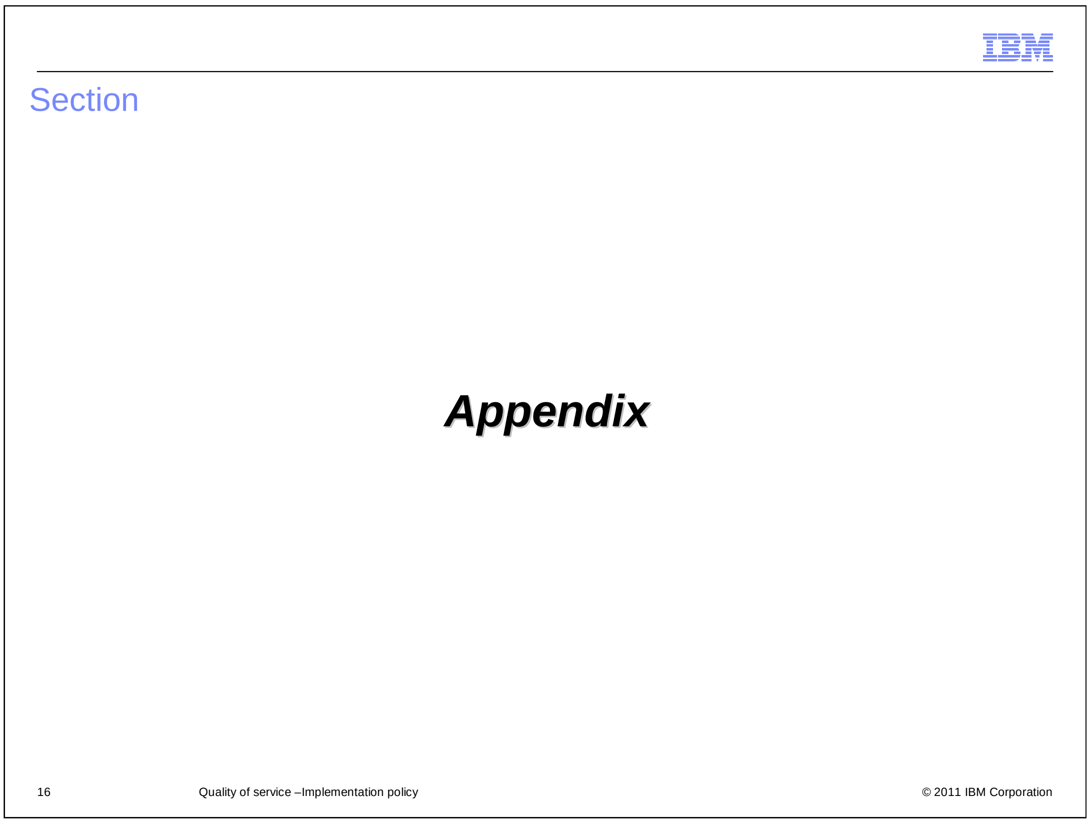


Summary

- Two types of implementation security policy: authorization and security identity.
- Used to protect SCA and to declare Run-as credentials
- You can limit or delegate access to components
- Mapping users or groups to a role authorizes access to applications by role
- Users, groups, and roles defined when application is installed or configured

15 Quality of service –Implementation policy © 2011 IBM Corporation

There are two types of implementation security policies, authorization and security identity. Authorization controls who can access the protected SCA resources. Security identity declares the security identity under which an operation is run. Both of these types are used to protect SCA components and operations and to declare the security identity under which the SCA components or operations run. You can limit access to an SCA component or to an operation to particular users or groups. You can also delegate access to another user when executing an SCA component or an operation. Different roles can have different security authorizations. Mapping users or groups to a role authorizes those users or groups to access applications defined by the role. Users, groups, and roles are defined when an application is installed or configured. The next several slides are an appendix that contain lists of supported intents and a list of old and new names for intents.

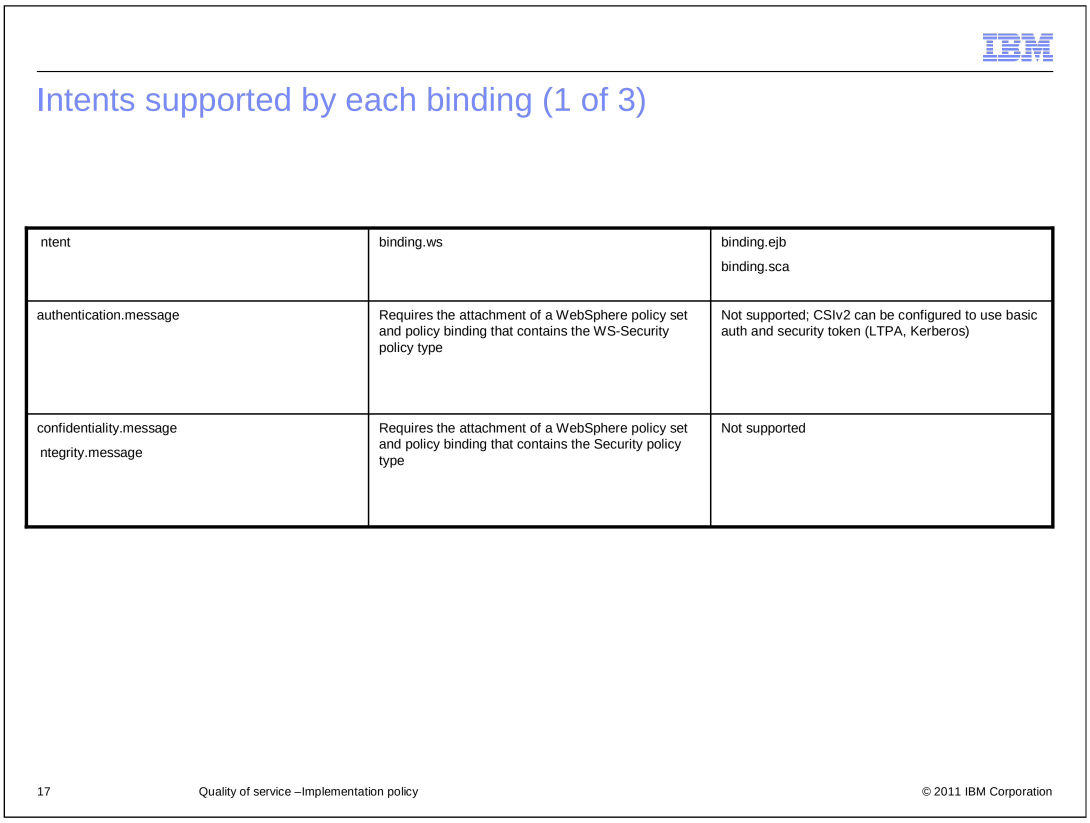


Section

Appendix

16 Quality of service –Implementation policy © 2011 IBM Corporation

This next section will show the appendix.

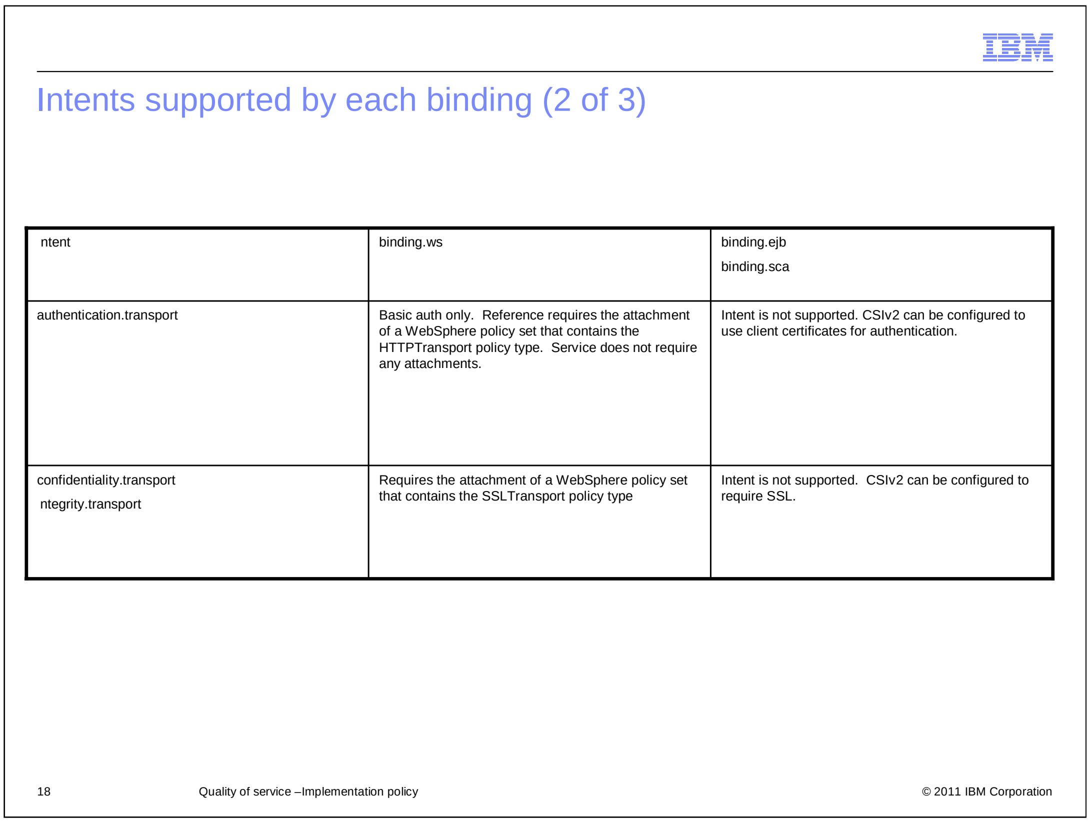


Intents supported by each binding (1 of 3)

ntent	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 ntegrity.message	Requires the attachment of a WebSphere policy set and policy binding that contains the Security policy type	Not supported

17 Quality of service –Implementation policy © 2011 IBM Corporation

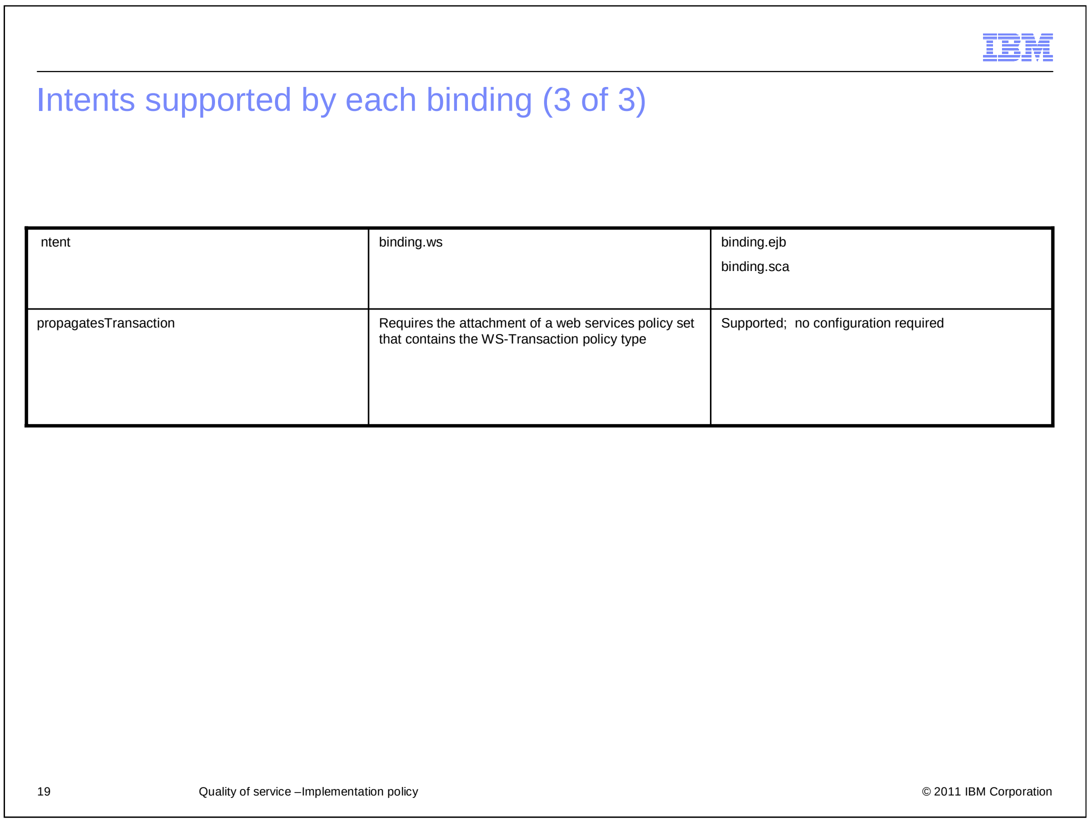
Intents supported by each binding.



Intents supported by each binding (2 of 3)

ntent	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 ntegrity.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.

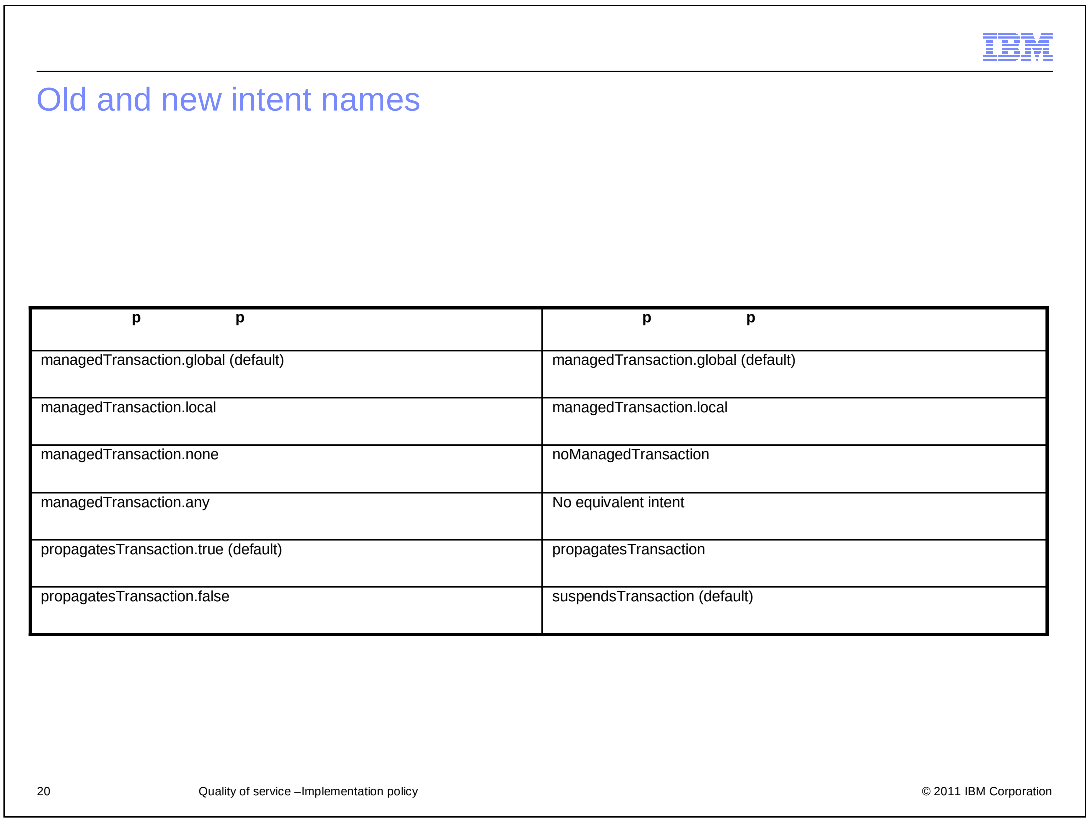
No notes to accompany slide.



Intents supported by each binding (3 of 3)

ntent	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

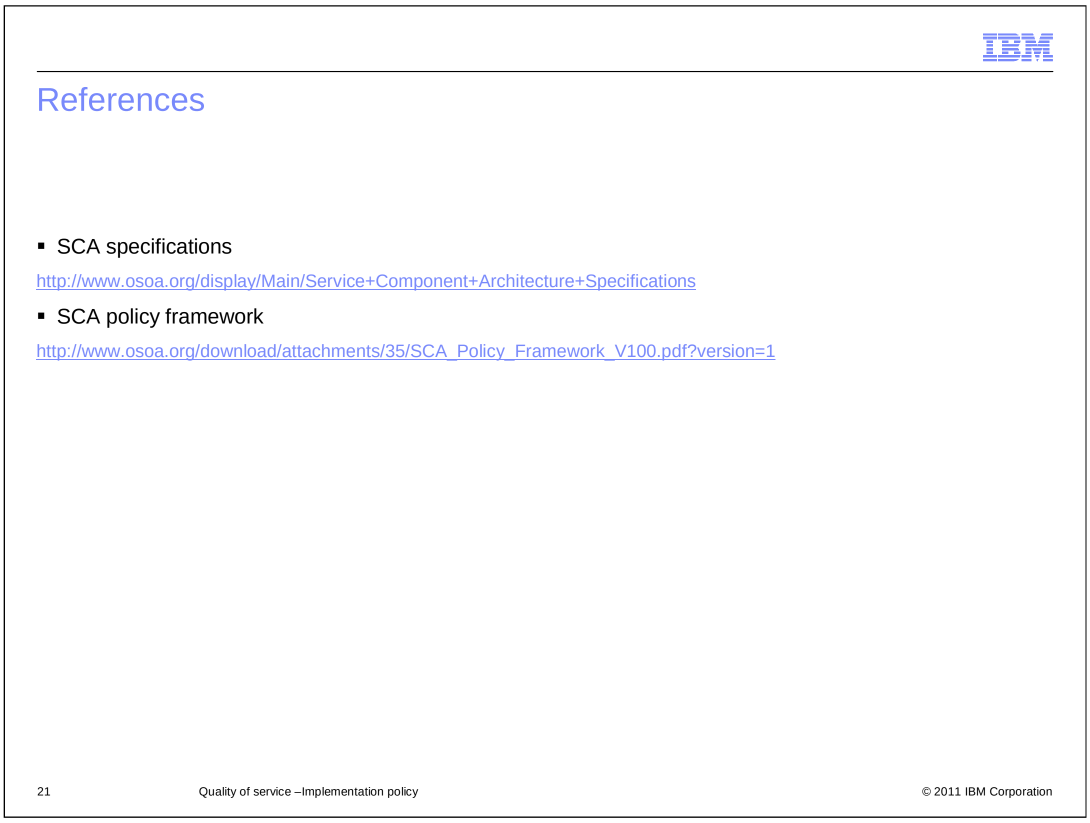
No notes to accompany slide.



Old and new intent names

p p	p p
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

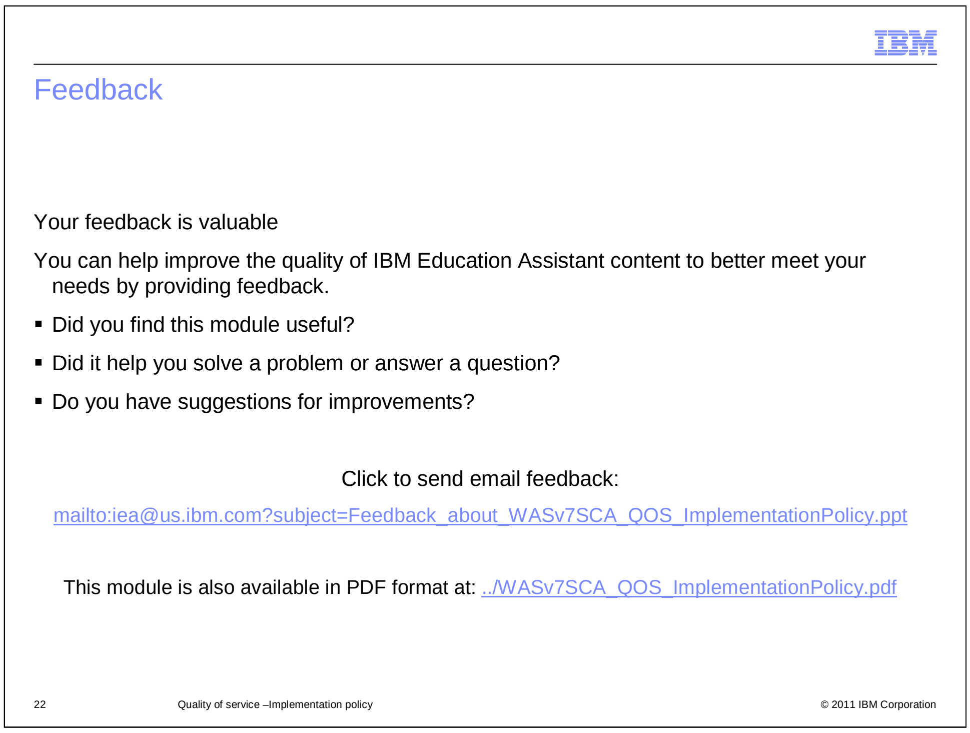
http://www.osoa.org/display/Main/Service+Component+Architecture+Specifications

- SCA policy framework

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

21 Quality of service –Implementation policy © 2011 IBM Corporation

This slide contains links of 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 email feedback:

mailto:iea@us.ibm.com?subject=Feedback_about_WASv7SCA_QOS_ImplementationPolicy.ppt

This module is also available in PDF format at: ../WASv7SCA_QOS_ImplementationPolicy.pdf

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

Trademarks, disclaimer, and copyright information