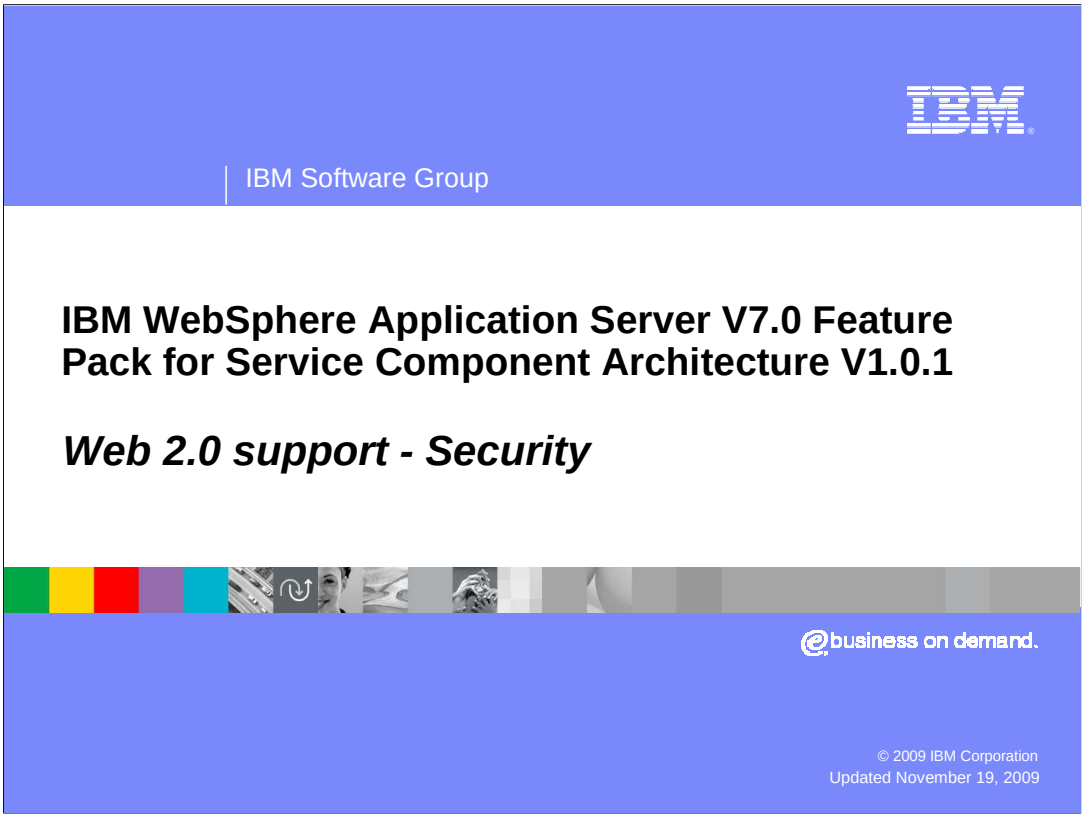
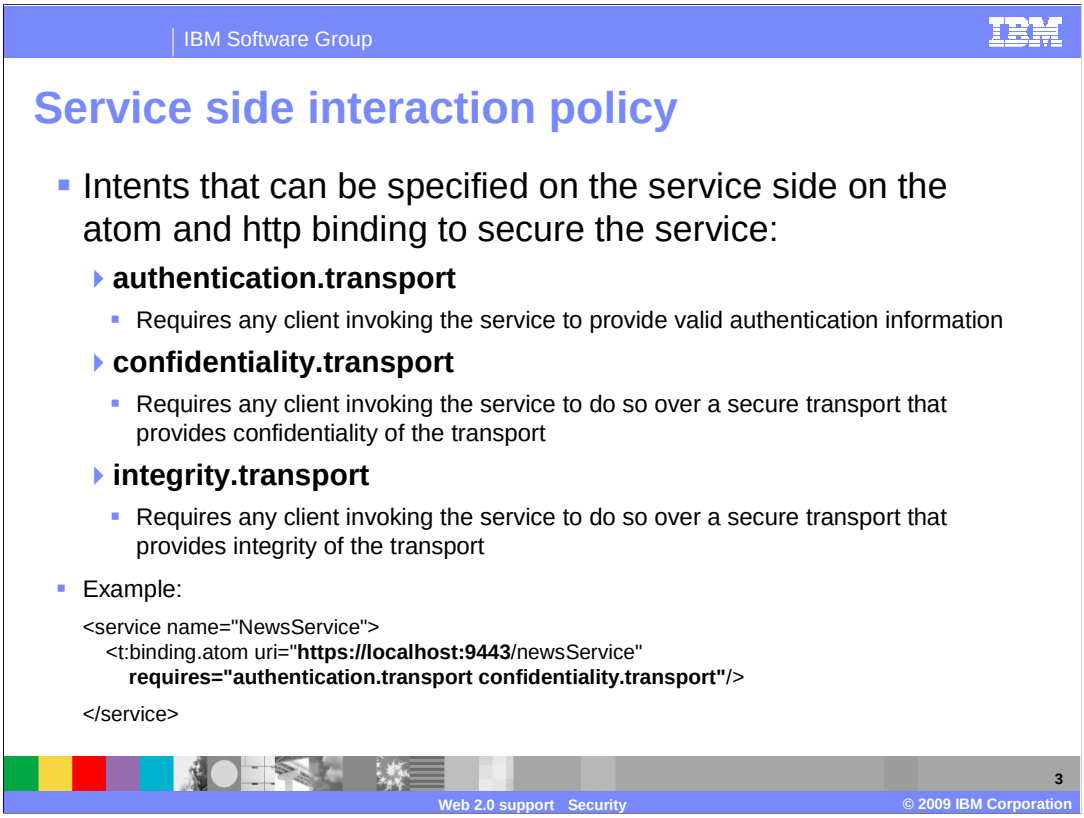


This presentation will talk about security with the Web 2.0 support in the Service Component Architecture feature pack.

Security

- Security interaction policy: authentication, confidentiality, integrity
  - As a Service Developer, you want to define a service using SCA and make it available over HTTP channel using Web 2.0 connectivity
    - a client must authenticate itself in order to use the SCA service
- Security implementation policy
  - Use SCA policySets on implementation.java
  - Not applicable to widget (HTML) as implementation
- Reliability policy
  - Not applicable

You can secure collections of data that are exposed by an Atom binding in an SCA application. An Atom binding can expose data as an Atom feed or reference existing external Atom feeds. As a Service Developer, you want to define a service using SCA and make it available over HTTP channel using Web 2.0 connectivity. You can do so such that a client must authenticate itself in order to use the SCA service or that assurance is required. It's required so that the contents of a message have not been tampered with and altered between sender and receiver. Security Implementation Policy can be implemented by using SCA policySets on implementation.java. You cannot apply security to the widget implementation implementation.widget. There is no reliability policy.



You can secure services that are exposed over an Atom and HTTP binding using intents. Administrative and application security must be enabled for the intents to be enforced. The three intents mentioned below are valid options for the **requires** attribute on the binding.atom element.

authentication.transport requires any client invoking the service to provide valid authentication information.

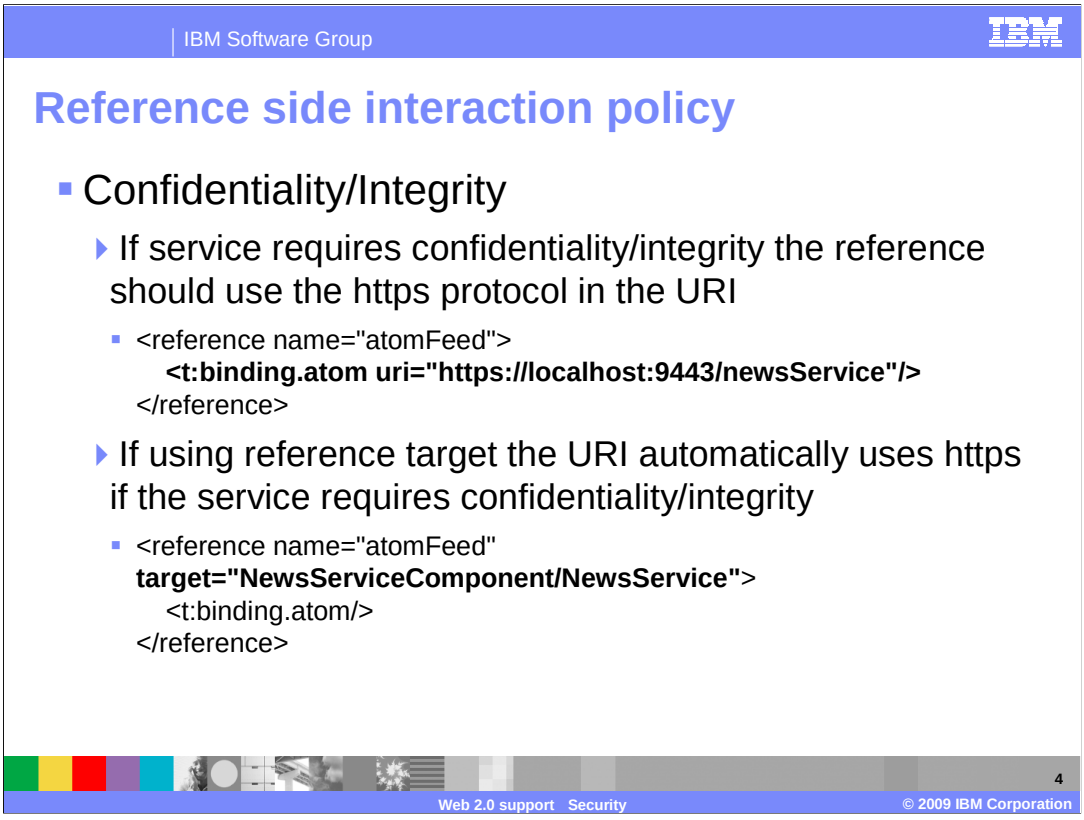
confidentiality.transport requires any client invoking the service to do so over a secure transport that provides confidentiality of the transport.

integrity.transport requires any client invoking the service to do so over a secure transport that provides integrity of the transport.

For example if you want to edit a composite definition that exposes a Java™ service over the Atom binding so that the exposed service requires a client to authenticate and communicate over a secure transport use;

```
<service name="NewsService">
   <t:binding.atom uri="https://localhost:9443/newsService"
      requires="authentication.transport confidentiality.transport"/>
 </service>
```

For more information on authorization policy, refer to documentation on SCA authorization and security identity policies in the information center.



If you want to invoke a secure service that is exposed over an Atom binding you can access the service directly from a browser or a client that supports Atom feeds. To access the feed directly, you can use the uniform resource indicator (URI) that the service specifies. If the service requires confidentiality or integrity, use the https protocol. If the service requires authentication, you are prompted by the browser to enter valid credentials. If a Java client is used to access the service, include the authentication information in the HTTP header.

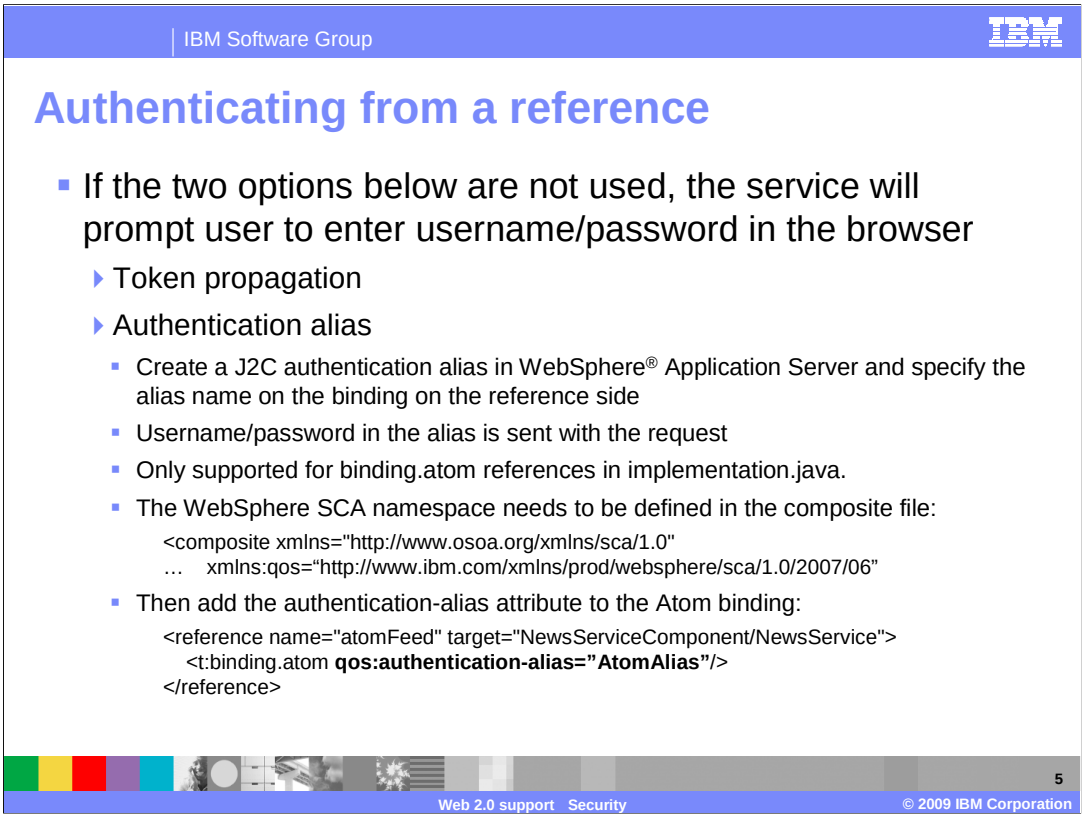
The example given invokes a service using a reference URI. If the service being referenced requires confidentiality or integrity, use the https protocol.

```
<reference name="atomFeed">
   <t:binding.atom uri="https://localhost:9443/newsService"/>
</reference>
```

You can also invoke the service using a reference target:

```
<reference name="atomFeed" target="NewsServiceComponent/NewsService">
   <t:binding.atom/>
</reference>
```

For this example, the invocation is secure if the service specifies the confidentiality.transport or integrity.transport intent.



When authenticating from a reference the service will prompt you to enter the user name and password in the browser if token propagation and authentication alias is not used. Token propagation occurs if there has been a successful authentication in the chain of calls before those credentials are propagated. This only works for service invocations in the same cell.

Authentication alias is a way to create a J2C authentication alias in WebSphere Application Server and specify the alias name on the binding on the reference side. The Username/password in the alias is sent with the request. This is only supported for binding.atom references in implementation.java.

Be sure to first create an authentication-alias using the administrative console or wsadmin commands. In the composite file, specify the alias name on the binding.atom element using the authentication-alias attribute. Be sure to define the WebSphere Application Server SCA namespace in the composite file too.

The example above shows the WebSphere Application Server SCA namespace and the authentication-alias being added to the Atom binding.

SCA authorization and security identity policies

- Protect SCA components and operations and declare the security identity under which the SCA components or operations are ran
  - An authorization policy controls who can access protected SCA components and operations
  - A security identity policy declares the security identity under which an SCA component or operation is executed
- Administration and Application security needs to be enabled in your WebSphere Application Server

There are two SCA declarative policies *authorization* and *security identity* to protect SCA components and operations and to declare the security identity under which the SCA components or operations are executed. An authorization policy controls who can access protected SCA components and operations. A security identity policy declares the security identity under which an SCA component or operation is executed. 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. Administration and application security in your WebSphere Application Server needs to be enabled.

Authorization for implementation.java

- Create a definitions.xml file and package it in the jar
- In your composite file define a namespace that matches the namespace used in the definitions.xml file

  <composite…xmlns:policy="http://testpolicy" …>
- Attach a policy set to the component implementation identifying the policySet in the definitions.xml:

  <implementation.java class="myClassImpl" **policySets="policy:allowRole1"**/>
- When adding the asset jar to the BLA map the roles in the policy set to real users/groups

```
<definitions
   xmlns="http://www.osoa.org/xmlns/sca/1.0"
   targetNamespace="http://testpolicy"
   xmlns:sca="http://www.osoa.org/xmlns/sca/1.0">
<policySet name="allowRole1"
   appliesTo="sca:implementation"
   xmlns="http://www.osoa.org/xmlns/sca/1.0">
   <authorization>
     <allow roles="staff"/>
   </authorization>
</policySet>
<policySet name="allowRole2"
   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>
```


Here are the steps to add SCA declarative policies *authorization* and *security identity*. First you will need to create a definitions.xml file and package this file in the jar. The definitions.xml file contains the policysets which specify the roles that have access. An example definitions.xml file is given on the right. Then, in your composite you will need to resolve the correct namespace for your policy within the root element of your composite file. This will need to match the namespace used in the definitions.xml file. To declare the policy namespace you will add the shown attribute into the composite tag

xmlns:policy=*"http://testpolicy"*

Next you will attach a policy set to the component implementation identifying the policySet that is defined in the definitions.xml file. For example you will add the shown attribute to your component implementation.java tag:

policySets=*"policy:allowRole1"*

Finally, when adding the asset jar to the business level application you need to map the roles in the policy set to real users/groups through the Map security roles to users or groups panel.

Development tools

- No specific runtime tool is required to deploy services provided or referred over Web 2.0 bindings
- No specific runtime tool is required to deploy HTML that is enhanced with SCA programming model
- Rational® Application Developer does not currently support new Web 2.0 connectivity
  - Atom binding, HTTP binding with wire format JSON-RPC , or Widget component implementation
  - One can edit the composite file directly to add these new Web 2.0 features

There is no specific runtime tools required to deploy services provided or referred over Web 2.0 bindings. There is no specific runtime tools required to deploy HTML that is enhanced with SCA programming model. Rational Application Developer does not currently support new Web 2.0 connectivity, that is Atom binding, HTTP binding with wire format JSON-RPC, or Widget component implementation access style in both its visual and textual composite editors. One can edit the composite file directly to add these new Web 2.0 features.

Section

Summary and references

The next section will provide a summary and references.

Summary

- You can secure collections of data that are exposed by an Atom binding in an SCA application
- You can secure services that are exposed over a Atom and HTTP binding using intents
  - Two SCA declarative policies *authorization* and *security identity* to protect SCA components and operations

You can secure collections of data that are exposed by an Atom binding in an SCA application. You can secure services that are exposed over a Atom and HTTP binding using intents. There are two SCA declarative policies *authorization* and *security identity* to protect SCA components and operations and to declare the security identity under which the SCA components or operations are executed.

References

- JSON: http://www.json.org
- Dojo Toolkit: http://dojotoolkit.org
- OpenAjax Alliance: http://openajax.org
- Ajax Technical library: http://www.ibm.com/developerworks/views/web/libraryview.jsp?search_by=Mastering+Ajax
- IBM education assistant: Feature pack for Web 2.0
  http://publib.boulder.ibm.com/infocenter/ieduasst/v1r1m0/index.jsp?topic=/com.ibm.iea.wasfpweb20/plugin_coverpage.html
- The WebSphere Application Server Feature Pack for Web 2.0 service page: http://www.ibm.com/software/webservers/appserv/was/featurepacks/web20/
- WebSphere Application Server Feature Pack for SCA service page: http://www-01.ibm.com/software/webservers/appserv/was/featurepacks/sca/
- Apache Tuscany: http://tuscany.apache.org/
  - Notice this presentation contains information from Apache Tuscany. You can find the license information here: http://www.apache.org/licenses/
- This article talks about authorization policy:

http://publib.boulder.ibm.com/infocenter/wasinfo/v7r0/index.jsp?topic=/com.ibm.websphere.soafep.multiplatform.doc/info/ae/ae/tsec_authsoa_policy.html

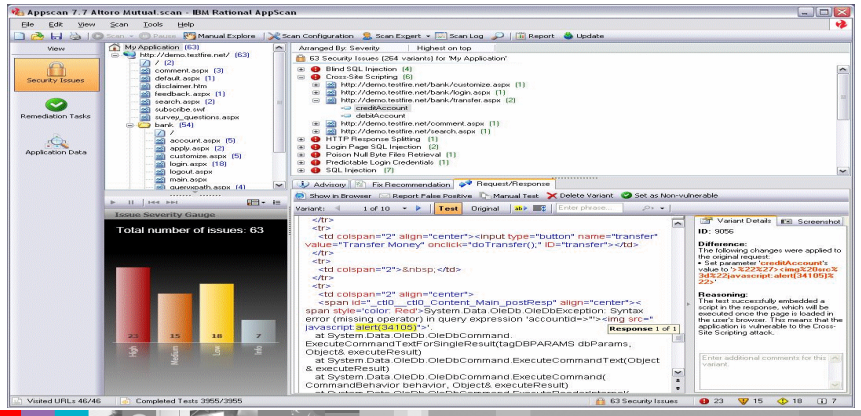
Above are some useful references.

IBM Software Group

Web 2.0 vulnerability testing

watchfire an IBM company

- Explicit Security vulnerability testing is critical when working with JavaScript/Web2.0
- Rational AppScan® (www.watchfire.com) is the industry-leading security vulnerability scanning tool for Ajax



12

Web 2.0 support Security

Outside-in security testing should be part of your quality assurance plan, watchfire.com is a security vulnerability testing tool.

Dojo Toolkit support

- For help with developing Ajax application look at the General Debugging Tools section
- The best information and help is on the www.dojotoolkit.org site especially the forums. Look through the forums for issues and post non-confidential issues on the forum.
- http://www.dojotoolkit.org/support
  - Frequently Asked Questions
  - Forums
- http://www.dojotoolkit.org/docs
  - The Book of Dojo 1.0
  - Dojo API Reference
  - Dojo Porting Guides

Here are some useful links for Dojo Toolkit.

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_Web20_security.ppt

This module is also available in PDF format at: ../WASV7SCA101_Web20_security.pdf

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

Trademarks, copyrights, and disclaimers