



IBM Software Group

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

### ***Web 2.0 support - Security***



@business on demand.

© 2009 IBM Corporation  
Updated November 19, 2009

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.

## Service side interaction policy

- Intents that can be specified on the service side on the atom and http binding to secure the service:
  - ▶ **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

- Example:

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



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.

## Reference side interaction policy

- Confidentiality/Integrity

- ▶ If service requires confidentiality/integrity the reference should use the https protocol in the URI

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

- ▶ If using reference target the URI automatically uses https if the service requires confidentiality/integrity

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



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.

## Authenticating from a reference

- If the two options below are not used, the service will prompt user to enter username/password in the browser
  - ▶ Token propagation
  - ▶ Authentication alias
    - Create a J2C authentication alias in WebSphere® Application Server and specify the alias name on the binding on the reference side
    - Username/password in the alias is sent with the request
    - Only supported for binding.atom references in implementation.java.
    - The WebSphere SCA namespace needs to be defined in the composite file:

```
<composite xmlns="http://www.osoa.org/xmlns/sca/1.0"
...  xmlns:qos="http://www.ibm.com/xmlns/prod/websphere/sca/1.0/2007/06"
```
    - Then add the authentication-alias attribute to the Atom binding:

```
<reference name="atomFeed" target="NewsServiceComponent/NewsService">
  <t:binding.atom qos:authentication-alias="AtomAlias"/>
</reference>
```

5

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

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

Web 2.0 support Security
© 2009 IBM Corporation

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](http://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](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](..WASV7SCA101_Web20_security.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](http://www.ibm.com), and the following terms are trademarks or registered trademarks of International Business Machines Corporation in the United States, other countries, or both:

AppScan IBM Rational 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>

Rational is a trademark of International Business Machines Corporation and Rational Software Corporation in the United States, Other Countries, or both.

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.

© Copyright International Business Machines Corporation 2009. All rights reserved.

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.

