IBM Software Group

IBM® WebSphere® Application Server V6

Security

Java™ Authorization Contract for Containers

This presentation will focus on Java Authorization Contract for Containers.

Goal

- Provide a overview of Java Authorization Contract for Containers (JACC) specification

2

The goals for this presentation are to provide an overview of Java Authorization Contract for Containers specification.

Prerequisite: Basic understanding of J2EE Security Model

Agenda

- J2EE Security Authorization model

- Java Authorization Contract for Containers (JACC) specification

- Tivoli® Access Manager (TAM) Client integration in WebSphere
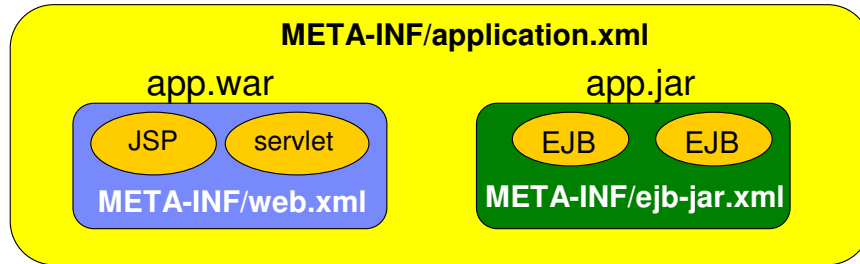
- JACC Configuration using Administration Console

3

The agenda for this presentation is listed in the above slide.

Section

J2EE Security Authorization Model

4

The next section will discuss the J2EE security authorization model.

J2EE Application: EAR file

META-INF/application.xml

app.war

JSP   servlet

META-INF/web.xml

app.jar

EJB   EJB

META-INF/ejb-jar.xml

- The web.xml is the deployment descriptor for the Web modules in the application and contains security-constraints

- The ejb-jar.xml is the deployment descriptor for the EJB modules in the application and contains method-permissions

- The application.xml is the deployment descriptor for the application and contains role definitions

5

The example shows the deployment descriptors of a J2EE application and its Web and EJB modules. The J2EE Security Roles are defined in the Application deployment descriptor. The authorization permissions by the J2EE Security roles are in the Web and EJB module deployment descriptor.

EJB Deployment Descriptor: Method Permission

```
<method-permission id="MethodPermission_1">
  <description>Manager access only </description>
  <role-name>Manager</role-name>
  <method id="MethodElement_1">
    <ejb-name>BankEJB</ejb-name>
     <method-name>getBalance</method-name>
    <method-params>
       <method-param>java.lang.Integer</method-param>
    </method-params>
  </method>
</method-permission>
```

- An example of the EJB deployment descriptor containing security policy information

- The getBalance method of BankEJB can only be accessed by the Manager Role

6

The EJB deployment descriptor shows the example of method called getBalance() that has been given permission to users that satisfy the J2EE Security Role of "Manager".

Role Definition and Binding: Application DD

application.xml

```
<security-role id="SecurityRole_1">
     <description>Manager in an enterprise</description>
     <role-name>Manager</role-name>
</security-role>
```

ibm-application-bnd.xmi

```
<authorizations xmi:id="RoleAssignment_1">
    <users xmi:id="User_1" name="Bob"/>
    <role href="META-INF/application.xml#SecurityRole_1"/>
    <groups xmi:id="Group_1" name="Managers"/>
</authorizations>
```

Manager **Role** has User "**Bob**" and Group "**Managers**"

7

The default binding of J2EE Security Roles to the users/groups is specified in the IBM Application Binding extension file. The binding is normally specified during application install. If no JACC provider is specified, this is how the binding information is stored. This is similar to V5 and also supported in V6.

Access Decision

Calling getBalance(1)

Request ───────────────────────────────▶ J2EE Server
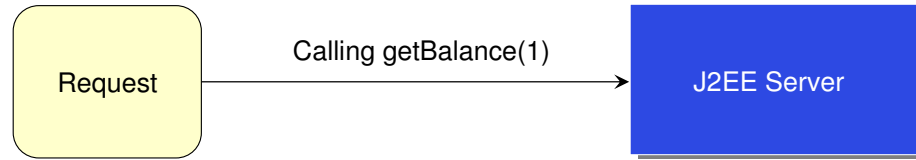
- Prompt the user to provide credentials (name/password)

- Check the credentials. If successful, create a Subject with the user information including the groups that the user belongs to

- Get the required roles for the getBalance(java.lang.Integer) method from the deployment descriptor.

- Get the assigned roles for the user from the binding file

- If the required roles match any assigned roles, access is permitted
  ▶ Otherwise denied

8

In the example shown here, the access decision steps are shown.

After a successful authentication of the client, authorization is checked before calling the method.

On authentication, a Subject is created for the client identity that has the information about the user and group, the user belongs. Then the roles of the user/group are determined from the role to user/group binding information. If the role matches with the required role for the method, access is granted. Otherwise, access is denied.

WebSphere Application Server Additions

- WebSphere Application Server supports two special subjects for J2EE applications:
  - ▶ **AllAuthenticated:** Subject implies any valid user in the User Registry
  - ▶ **Everyone:** Subject implies any user

9

Besides the normal binding of the roles to the users and groups, WebSphere Application Server supports 2 special subjects, "**All Authenticated" and "Everyone".**

The J2EE roles for the applications can be bound to the special subjects.

Binding to Everyone gives access to all users, whether they are in the user registry or not.

Binding to All Authenticated gives access to valid users in the User registry.

Section

JACC Specification

The next section will discuss the Java Authorization Contract for Containers Specification.

JACC Introduction

- JACC allows applications servers to interact with 3rd party authorization providers using standard interfaces to make authorization decisions
  - ▸ JACC defines permission classes for both the EJB and Web container

- JACC does not specify how to assign principals to roles

11

Support of JACC based Authorization provider in WebSphere Application Server V6 is in addition to the Default Authorization, using the IBM Binding file for authorization information.

JACC allows authorization information (J2EE Security roles to user/group binding) in an external JACC providers.

JACC Example

WebSphere Application Server V6

Application Installation

JACC Provider Contract

PolicyConfiguration

Provider Repository
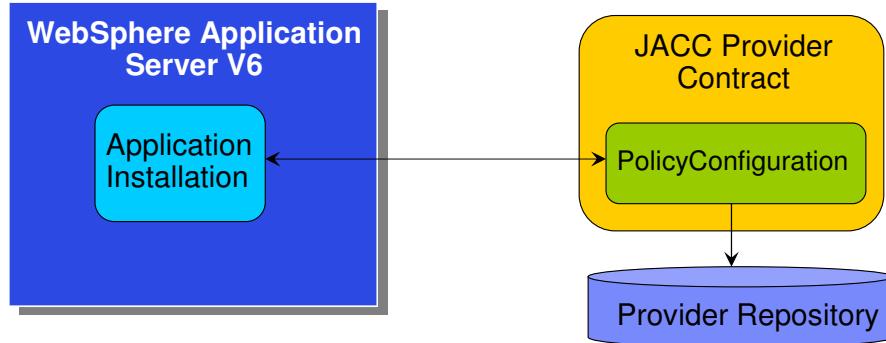
- Create contextID unique to the module being installed
- Get PolicyConfiguration Object for the contextID
- Propagate security policy information for the module using the PolicyConfiguration Object

The slide shows an example of how the Application Server interacts with a 3rd party JACC provider. During application installation, information on the Security binding is send to the JACC provider. During checking of the authorization permission, the Application Server queries the JACC provider to get the roles associated with the user/group.

Deploying an Application Using JACC

- During application installation, translate the security policy in the deployment descriptor to the appropriate permission objects

- Associate the permission objects with the appropriate roles

- Create a unique identity (contextID) for the module being deployed

- Propagate the information to the provider using the PolicyConfiguration object implemented by the provider

- Link all the modules in an application and commit

13

IBM

Application Server Container Requirements

WebSphere
Application Server V6

Provider Repository

Access J2EE
resource

EJB/Web
Container

Check
access

Policy Object

yes/no

yes/no

JACC Provider
Contract
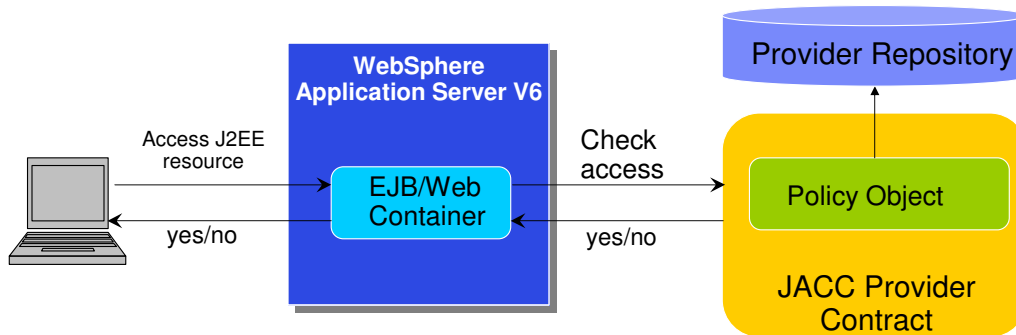
- Create contextID for the module being accessed
- Create the appropriate Permission object for the resource
- Register information required by the specification
- Delegate the access decision to the Policy object

Java Authorization Contract for Containers

14

© 2005 IBM Corporation

Application Server Container Requirements

- Authenticate the user by checking the user's credentials

- Create a permission object for the resource being accessed

- Register required information by using the PolicyContextHandler objects

- Create the unique identity for the module being accessed

- Call the java.security.Policy object implemented by the provider to make the access decision

15

JACC Configuration in Administrative Console

Show Me of JACC Configuration

The Administrative Console user interface to set the Authorization providers is shown here.

Can either use the Default Authorization, where the role to user/group binding is in the IBM application binding file, or can use an external JACC provider.

The external JACC provider configuration panel is pre-filled for values to be used by Tivoli Access Manager as the JACC provider. When using another JACC provider, the fields will have to modified for that JACC provider.

Click the Show-me icon for a demonstration on how to configure 3rd party JACC provider, including TAM.

Section

Tivoli Access Manager (TAM) Integration

The next section will discuss Tivoli Access Manager integration.

TAM Introduction

- Provides unified authentication and authorization services for heterogeneous environments

- Enforces security by using a single security policy server across multiple file types, application providers, devices and protocol

- Access decisions are based on information held external to the application

18

TAM Components

WebSphere Application Server V6

Access Manager for WebSphere (AMWAS) Component

| TAI | JACC Provider Contract | JACC Management | GSO Credential Mapping |

Access Manager Java Runtime (AMJRTE) Component

| PDJAdmin (Management) | Local ACL DB Replica | PDPerm (Authz) | PDPrincipal (Authentication) |

SSL Connections

Access Manager Policy Server

User Registry

Master ACL DB

AM Authz Server

ACL DB Replica

Access Manager Server

TAM Integration

- TAM client pieces are embedded in the WebSphere Application Server V6

- TAM is the default JACC provider for WebSphere

- TAM server is included with WebSphere Application Server V6 Network Deployment package

- TAM client can be configured using the scripting or the Administration Console

- In addition to authorization, TAM server can also provide authentication functionality

20

© 2005 IBM Corporation

The TAM client is embedded within the WebSphere Application Server V6. TAM is the default JACC provider.

The Administrative Console panels for an external JACC provider is pre-filled with values needed for TAM as the default external JACC provider.

You can use other JACC providers and configure them within the Administrative console.

Besides being a JACC provider for authorization, TAM can also be used for authentication.

Advantages using TAM

- Industry leading security provider

- Supports account and password policies
  ▸ For example, 3 strikes you are out policy

- Supports dynamic changes to the authorization table without having to restart the applications

- Tight integration with WebSphere Application Server V6

Here, some of the advantages of using TAM as JACC provider, are listed. WebSphere Application Server V6 has tighter integration with TAM than previous releases. The authorization table can be modified, and that does not require that the application be restarted. During authorization queries, the updated information will be retrieved.

Section

Summary and References

22

The next section will discuss a summary of the aforementioned concepts.

Summary and References

- JACC allows application servers to interact with third party authorization providers using standard interfaces

- TAM provides unified authentication and authorization services for heterogeneous environments
  - ▸ TAM client pieces are embedded in the WebSphere Application Server
  - ▸ TAM server is bundled in the WebSphere Application Server V6 Network Deployment package
  - ▸ TAM is the default JACC provider for WebSphere

23

Java Authorization Contract for Containers

© 2005 IBM Corporation

In summary, this presentation has focused on how JACC allows application servers to interact with third party authorization providers via standard interfaces, as well as how TAM provides unified authentication and authorization services for heterogeneous environments.

Trademarks, Copyrights, and Disclaimers