SAP NetWeaver 2004s SPS 4

Security Guide

Security Guide for Connectivity with the J2EE Engine

Document Version 1.00 – October 24, 2005

THE BEST-RUN BUSINESSES RUN SAP

SAP AG
Neurottstraße 16
69190 Walldorf
Germany
T +49/18 05/34 34 24
F +49/18 05/34 34 20
www.sap.com

These materials are subject to change without notice. These materials are provided by SAP AG and its affiliated companies ("SAP Group") for informational purposes only, without representation or warranty of any kind, and SAP Group shall not be liable for errors or omissions with respect to the materials. The only warranties for SAP Group products and services are those that are set forth in the express warranty statements accompanying such products and services, if any. Nothing herein should be construed as constituting an additional warranty.

Disclaimer

Some components of this product are based on Java™. Any code change in these components may cause unpredictable and severe malfunctions and is therefore expressively prohibited, as is any decompilation of these components.

Any Java™ Source Code delivered with this product is only to be used by SAP's Support Services and may not be modified or altered in any way.

Documentation in the SAP Service Marketplace

You can find this documentation at the following Internet address: `service.sap.com/securityguide`

Typographic Conventions

Type Style	Description
Example Text	Words or characters quoted from the screen. These include field names, screen titles, pushbuttons labels, menu names, menu paths, and menu options. Cross-references to other documentation
Example text	Emphasized words or phrases in body text, graphic titles, and table titles
EXAMPLE TEXT	Technical names of system objects. These include report names, program names, transaction codes, table names, and key concepts of a programming language when they are surrounded by body text, for example, SELECT and INCLUDE.
`Example text`	Output on the screen. This includes file and directory names and their paths, messages, names of variables and parameters, source text, and names of installation, upgrade and database tools.
`Example text`	Exact user entry. These are words or characters that you enter in the system exactly as they appear in the documentation.
`<Example text>`	Variable user entry. Angle brackets indicate that you replace these words and characters with appropriate entries to make entries in the system.
`EXAMPLE TEXT`	Keys on the keyboard, for example, *F2* or *ENTER*.

Icons

Icon	Meaning
⚠	Caution
🗨	Example
💡	Note
🧭	Recommendation
SYN	Syntax

Additional icons are used in SAP Library documentation to help you identify different types of information at a glance. For more information, see *Help on Help* → *General Information Classes and Information Classes for Business Information Warehouse* on the first page of any version of *SAP Library*.

Contents

Security Guide for Connectivity with the J2EE Engine

J2EE Connector Architecture Security

The J2EE Connector Architecture (JCA) enables connectivity to back-end systems such as Enterprise Information Systems (EIS), using resource adapters. The adapters are modules that are deployed on a J2EE compatible application server and provide unified access to the resource system for any application components that are also installed on the server.

For information about how to implement security functions when using the resource adaptors, see Implementing Security Functions [SAP Library].

Remote Method Invocation

Java Remote Method Invocation (RMI) enables the creation of Java-to-Java applications between remote Java Virtual Machines (JVMs). An example of the use of RMI is the communication between external Java applications and the SAP Web AS.

You can use RMI either using the P4 protocol, which is an SAP-proprietary protocol, or the Internet Inter-ORB Protocol (IIOP).

The security aspects involved when using RMI are described in the following topics:

- Authentication for RMI-P4 Clients [Page 5]
- Using P4 Protocol Over a Secure Connection [Page 6]
- Security for RMI-IIOP Applications [Page 7]
- Configuring the J2EE Engine for IIOP Security [Page 8]

See also:

Connectivity and Interoperability [SAP Library]

1 Authentication for RMI-P4 Clients

RMI-P4 clients authenticate themselves to the naming system on the J2EE Engine when the InitialContext is obtained. The authentication is performed using the `BASIC` authentication scheme, that is, by username and password. The client's identity is checked against the security role settings from the naming system policy configuration in the Security Provider Service to determine whether it can obtain the `InitialContext`. By default, all users of the J2EE Engine can obtain the `InitialContext` and perform lookup operations from it.

Providing the Client Credentials

The username and password of the RMI-P4 client are provided as environment properties in the source code when the InitialContext is obtained. The variables that must be used are `javax.naming.Context.SECURITY_PRINCIPAL` with the username as the value, and `javax.naming.Context.SECURITY_CREDENTIALS` with the user password as the value.

Propagating the Client Credentials to the Server-side Remote Objects

If the server-side remote objects define their own security requirements, the RMI-P4 client credentials available to the `InitialContext` are propagated to them in order to determine access rights to business methods. The server-side remote objects can be:

- Enterprise beans

  These define role-based access restrictions using the EJB application's deployment descriptors.

- Java classes that implement remote interfaces

  These must define access restrictions using the appropriate APIs in their own code.

2 Using P4 Protocol Over a Secure Connection

Use

You can set up a secure connection for P4 communication. You can lay the P4 messages over both SSL and HTTPS protocols.

Prerequisites

- You configured your J2EE Engine to use SSL. For more information, see Configuring the Use of SSL on the J2EE Engine [SAP Library].
- You configured the port for secure connections that the P4 Provider Service running on the Java dispatcher listens to. Then, the client uses that port to open the secure connection in its request to the J2EE Engine.

  For more information, see Managing the Underlying Transport Layers [SAP Library].

Procedure

In your client code, obtain the `InitialContext` to connect to the remote object using the following properties:

1. Specify the port for the secure connection with the provider URL property.
2. Specify the underlying transport layer you want to use. You use the `TransportLayerQueue` property with value SSL for P4 over an SSL connection, or HTTPS for P4 over an HTTPS connection.

Example

Below is an example of client code that obtains `InitialContext` using a specific transport layer. This transport layer (SSL or HTTPS) is specified by the `transportType` parameter provided in the command line when the following client code is executed:

```
public void init(String host, String port, String user, String pass,
String transportType)
      {
            try
            {
                  Properties p = new Properties();
                  p.put("java.naming.factory.initial",
"com.sap.engine.services.jndi.InitialContextFactoryImpl");
                  p.put("java.naming.provider.url", host + ":" + port);
                  p.put("java.naming.security.principal", user);
                  p.put("java.naming.security.credentials", pass);
                  // The transportType parameter has value ssl or
https.
         // It is provided on the command line.
         p.put("TransportLayerQueue", transportType);
                  ctx = new InitialContext(p);
                  System.out.println("NamingClient.run1 ctx : " + ctx);
            }
            catch(NamingException e)
            {
                  System.out.println(">> Exception : " +
e.getMessage());
                  e.printStackTrace();
            }
      }
```

3 Security for RMI-IIOP Applications

Security aspects for RMI-IIOP applications are defined by the Common Secure Interoperability V2 Specification. The J2EE Engine's Object Request Broker (ORB) implementation fully supports conformance level 0 of this specification. The client-side ORB must also implement this specification so that the client can use the various security functions for executing methods on the remote objects.

You can make use of the following security aspects in your RMI-IIOP applications:

- Transport layer security

  You can require that the messages transport is conducted over an SSL layer to ensure data integrity and confidentiality. Also, you can specify the handshake procedure to be used – one- or bi-directional.

- Authentication layer security

  You can specify the authentication mechanisms to be used for user authentication and the realm that the client credentials are valid for. The J2EE Engine's ORB currently supports authentication by username and password only.

- Caller identity propagation

  Specifies whether caller identity assertion is supported.

All these security aspects are controlled by the application developer. This means that he or she configures the requirements for the server-side application using the deployment descriptors (in the case of EJB applications), or handles the task programmatically in the remote objects code. The client, on the other hand, uses the appropriate methods provided by the client-side ORB accordingly to authenticate itself to the server-side application and get access to its business methods.

For more information about the security aspects you can configure for your server-side EJB applications, see Specifying Security When Using IIOP [SAP Library].

In order to use security for RMI-IIOP applications, you must first configure the J2EE Engine [Page 8].

4 Configuring the J2EE Engine for IIOP Security

Use

So that you can use security for your RMI-IIOP applications, you need to provide an additional parameter for the Java Virtual Machine that can be used to initialize the J2EE Engine's ORB.

Prerequisites

You must have configured your J2EE Engine to use SSL if you want to use SSL to encrypt IIOP messages.

Procedure

We assume you have stopped the J2EE Engine; to configure it for IIOP security, proceed as follows:

1. Start the offline configuration editor tool provided with your J2EE Engine installation. Use the *offlinecfgeditor* script file located in the *<SAP_install_dir>\<system_name>\<instance_name>\j2ee\configtool* directory.

2. In the *Configurations* tree that appears on the screen, browse to the *cluster_data* → *Propertysheet instance.properties.ID<number>*.

3. To switch the editor from view to edit mode, choose ✎. Since you have stopped your J2EE Engine, you can choose *Yes* when the *Switch to Edit* mode warning appears.

4. Select the *Propertysheet instance.properties.ID<number>* and choose ✎ to open it for editing.

   A *Change Configuration* screen appears.

5. In the name column, find the key *ID<number>.JavaParameters* that defines the parameters of the JVM in which the server instance runs.

   You can locate the ID number of the server process by locating the property with name *ID<number>.Name* and value of *server*.

   Select the *ID<number>.JavaParameters* field and a new *Change property entry* screen appears.

6. Enter the following parameter in the *Custom:* field:

   ```
   -Dorg.omg.PortableInterceptor.ORBInitializerClass.
   com.sap.engine.services.iiop.csiv2.interceptors.SecurityInitialize
   r
   ```

7. Choose *Apply custom* to save your custom parameters. Then choose *OK* from the *Change Configuration* screen.

8. Exit the Config Tool.

9. Start your J2EE Engine.

Result

The J2EE Engine must now be appropriately initialized to support security for RMI-IIOP applications.