RMI Clients on SAP NetWeaver

SAP Platform Ecosystem

Copyright

Icons in Body Text

Icon	Meaning
	Caution
	Example
	Note
	Recommendation
	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*.

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.

Table of Contents

Preface

The Java Remote Method Invocation (Java RMI) framework enables the java developer to develop distributed applications in a robust and effective way. The communication between Java applications that run in different Java virtual machines on different hosts can be easily achieved with RMI. By using RMI calling a remote object's method is almost as simple as working in a non distributed environment. To get a deeper understanding of RMI the Java RMI specification and the Tutorials and Product Documentation provided by Sun are good starting points.

This document intends to give you a quick practical introduction on using RMI with the SAP Web Application Server focusing on the client side. By importing documented demo projects into your SAP NetWeaver Developer Studio you will learn exemplary, what it programmatically takes to communicate with the server. Beside that the document explains the demo projects, points you to the things you have to care of and guides you to further information. Using RMI to write a GUI-Client for instance with Swing, a plain Console Client or simply unit tests that check your business logic is what you learn out of the tutorial.

Prerequisites

Systems, installed applications, and authorizations

- ☐ SAP Web AS – Java 6.40

Knowledge

- ☐ Basic understanding of Java Remote Method Invocation (RMI)

RMI Clients

Short introduction to RMI-P4

Distributed object applications are required to locate and communicate with distributed objects and load classes of objects that are transmitted during the communication. The objects are located by J2EE Engine's Naming System (JNDI implementation), instead of a classic RMI registry. The communication is done by the SAP proprietary P4 protocol. The class loading is done by RMI-P4 which has less limitation than the standard RMI implementation. RMI-P4 allows dynamic class loading, offers an effective garbage collection mechanism, load balancing and a failover mechanism.

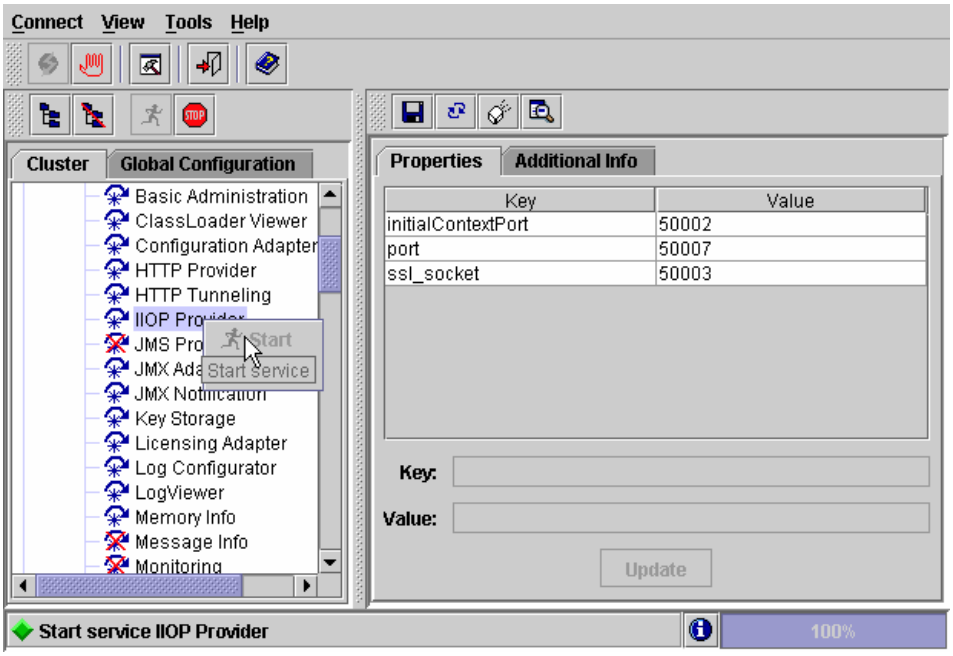
With RMI-P4 you can use two different Stubs and Skeletons types. You can generate Stubs and Skeletons with the J2EE Engine's rmic tool or alternatively generate the Stubs and Skeletons dynamically on runtime. Dynamic generation releases the developer of the duty to generate stubs and skeleton. Caused by the additional expense of the dynamic generation it is slower than the rmic approach.

Short introduction to RMI-IIOP

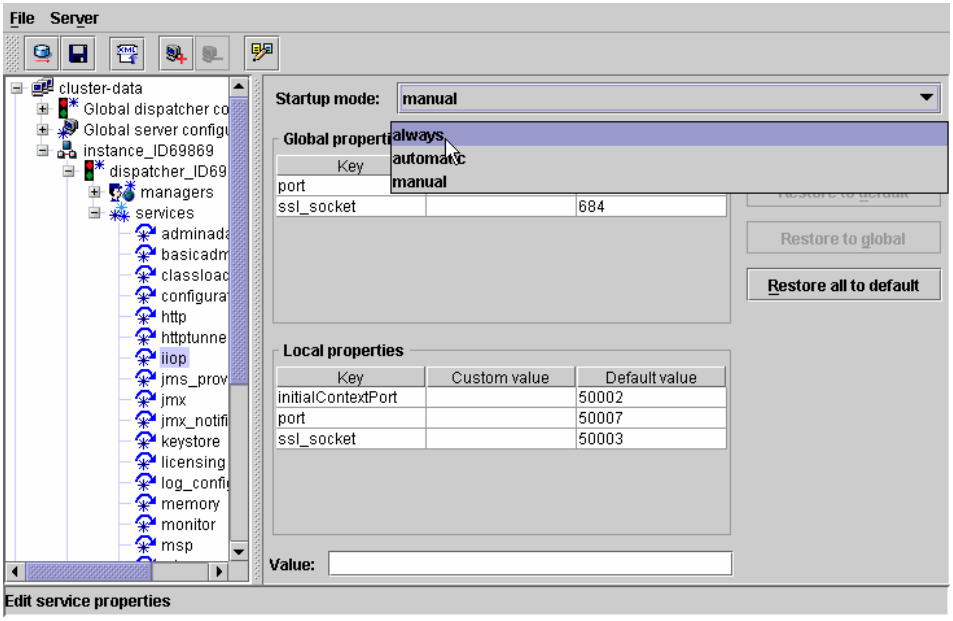
By using the communication protocol of CORBA Internet Inter-ORB Protocol (IIOP). RMI-IIOP applications are enabled to communicate with Java and CORBA clients. Since CORBA is language independent it is for instance possible to communicate between a RMI-IIP and C++ client.

Configuration of the J2EE Engine

For the usage the protocols P4 and IIOP the corresponding Provider Services must run on **the dispatcher and the server**. By default the services for IIOP won't be started. The P4 Provider Service and IIOP Provider Service can be manually started with the Visual Admin. The services can be configured to start always on startup of the server with Config Tool. The following pictures demonstrate the configuration.



Picture 1 Starting IIOP Provider Service with Visual Admin



Picture 2 Configure IIOP Provider Service to start always on startup with the Config Tool

Importing the Initial Projects

To focus the development of this example application to the actual content covered, there is a predefined sample application template available in the SAP Developer Network (SDN) http://sdn.sap.com (specify the taxonomy in more detail!).

Prerequisites

You have access to the SAP Developer Network (http://sdn.sap.com) with a user ID and password.

☐ The SAP NetWeaver Developer Studio is installed on your computer.

Procedure

Importing the initial projects into the Developer Studio

1. Call the SAP NetWeaver Developer Network using the URL http://sdn.sap.com and log on with your user ID and the corresponding password. If you do not have a user ID, you must register before you can log on.
2. Navigate to *Web Application Server* area and then to the *Samples and Tutorials* section.
3. Download the ZIP file *TutFatClientsWithWebStart.zip*
4. Unzip the contents of the ZIP file into the work area of the SAP NetWeaver Developer Studio or in local directory.
5. Call the SAP NetWeaver Developer Studio.
    a. To do this, choose *File* → *Import* in new menu.
    b. In the next window choose *Multiple Existing Projects into Workspace* and choose *Next* to confirm.
    c. Choose *Browse*, open the folder in which you unzipped the projects *TutFatClientsWithWebStart.zip*
    d. Select the all projects of the ZIP-File, check the the *open* checkbox and choose *Finish* to confirm.

Getting the InitialContext for JNDI lookups

The different scenarios for using RMI on the client side have in common, that the server objects are found by a JNDI lookup. Therefore this chapter gives you a short introduction how to configure the JNDI properties to get access to the server.

Table 1 JNDI properties for RMI-P4

key	value
Context.INITIAL_CONTEXT_FACTORY	com.sap.engine.services.jndi.InitialContextFactoryImpl
Context.PROVIDER_URL	<host>:<p4 port> (J2EE Engine Ports)
Context.SECURITY_PRINCIPAL	<username>
Context.SECURITY_CREDENTIALS	<password>

Table 2 JNDI properties for RMI-IIOP

key	Value
Context.INITIAL_CONTEXT_FACTORY	com.sap.engine.services.jndi.CosNamingInitialContextFactoryImpl or com.sun.jndi.cosnaming.CNCtxFactory
Context.PROVIDER_URL	<host>:<iiop port> (J2EE Engine Ports)
Context.SECURITY_PRINCIPAL	<username>
Context.SECURITY_CREDENTIALS	<password>

The demo projects

The demo projects provide the infrastructure for the communication between the client and the server. The simple demo scenario is the following.

1. Client connects to the server.
2. Client calls Server's method *sayMyName* with name as parameter
3. Server returns a Greeting String to the client.

3 projects are needed for the scenario:

- TutWritingRMIClients_EJB (containing an EJB which acts as the server)
- TutWritingRMIClients_EAR (Enterprise application project for deployment of the EJB)
- TutWritingRMIClients_Client (The Client)

In the following chapters the server and client side is briefly described.

The Server Side

A stateless Session Bean acts as a server containing one business method called *sayMyName.*

```
com.example.ExampleBean
...

public class ExampleBean implements SessionBean {
...
   public String sayMyName(String myName) {
```

```
      return "Hello " + myName + ".";
   }
...
}
```

The Client Side

The client applications for RMI-P4 and RMI-IIOP do the following:

4. Locate the home interface for the bean.
5. Get a reference to the component interface from the home interface
6. Use the component interface to call a business method of the bean.

The following source code shows, how this can be done for a RMI-P4 Client.

com.example.rmi.clients.EjbClientP4

```
...
public class EjbClientP4 {
   /**
    * Calls a business method of an EJB by using RMI-P4.
    */
   public static String sayHello_P4(String myName) {
      // The String which will be returned to the Client
      String result = "";
      try {
         // Get the InitialContext for the JNDI lookup
         Context ctx = new InitialContext(JndiHelper.getP4JndiProps());
         //Lookup the bean and cast to remote home interface
         ExampleHome home =
            (ExampleHome) ctx.lookup(
               "example.com/TutWritingRMIClients_EAR/ExampleBean");
         // Get reference to the component interface
         Example bean = home.create();
         // Invoke business method
         result = bean.sayMyName(myName);
      } catch (Exception e) {
         e.printStackTrace();
      }
      return result;
   }
   public static void main(String[] args) {
      // Execute the method
      System.out.println(sayHello_P4("P4"));
   }
}
```

The RMI-IIOP Client looks almost the same. The only differences are the different Properties for the JNDI lookup and the need to narrow the object returned of the lookup.

com.example.rmi.clients.EjbClientIIOP

```
...
         // Get the InitialContext for the JNDI lookup
         Context ctx = new InitialContext(JndiHelper.getIIOPJndiProps());
         // Lookup the bean
         Object obj =
   ctx.lookup("example.com/TutWritingRMIClients_EAR/ExampleBean");
         // Narrow and cast to remote home interface
```

```
         ExampleHome home =
            (ExampleHome) PortableRemoteObject.narrow(
               obj,
               ExampleHome.class);
...
```

You may have to adjust the client code to your system environment for the JNDI lookups. This can be done by editing the following constants in the described class.

```
com.example.rmi.clients.util.JndiHelper
...
public class JndiHelper {
   //TODO Adjust the following values to your configuration.
   private final static String HOST = "localhost";
   private final static int P4_PORT = 50004;
   private final static int IIOP_PORT = 50007;
   private final static String PRINCIPAL = "Administrator";
   private final static String CREDENTIALS = "admin";
...
}
```

Running the RMII-P4 Client

Prerequisites

1. JNDI Properties are set according to your environment

2. In the client's classpath there must be:

- Bean's remote home interface
- Bean's remote component interface
- sapj2eeclient.jar
- logging.jar
- exception.jar

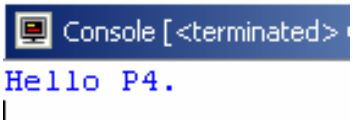
Procedure

1. Deploy the TutWritingRMIClients_EAR.ear
   a. Start the *SAP NetWeaver Application Server*
   b. Start the *SAP NetWeaver Developer Studio*
   c. Open the *J2EE Development Perspective* and chose the *J2EE Explorer View*
   d. Chose in the context menu of the *TutWritingRMIClients_EAR* project *Build Application Archive*
   e. Chose in the context menu of the *TutWritingRMIClients_EAR.ear Deploy to J2EE Engine*

2. Run *com.example.rmi.clients.EjbClientP4* in *TutWritingRMIClients_Client*
   a. Open the *Java Perspective* and chose the *Package Explorer View*
   b. Open the *src*-folder
   c. Open the *com.example.rmi.clients*-package
   d. Double click on *EjbClientP4.java*
   e. Select in the Menu Run/Run As/Java Application

Result

In the Console View the following Output appears in case of a successful execution of the client:



Running the RMII-IIOP Client

Prerequisites

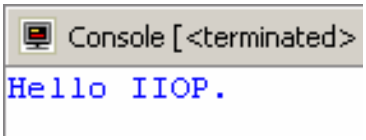
1. JNDI Properties are set according to your enviroment
2. IIOP Provider Service must run on Dispatcher and Server
3. In the client's classpath there must be:
   - The Client Jar File (Getting a Client Jar)
   - Bean's remote home interface
   - Bean's remote component interface
   - sapj2eeclient.jar
   - logging.jar
   - exception.jar

Procedure

1. Deploy the TutWritingRMIClients_EAR.ear
   a. Start the *SAP NetWeaver Application Server*
   b. Start the *SAP NetWeaver Developer Studio*
   c. Open the *J2EE Development Perspective* and chose the *J2EE Explorer View*
   d. Chose in the context menu of the *TutWritingRMIClients_EAR* project *Build Application Archive*
   e. Chose in the context menu of the *TutWritingRMIClients_EAR.ear Deploy to J2EE Engine*
2. Enable IIOP Support for the EJB Application
3. Run *com.example.rmi.clients.EjbClientIIOP* `in` *`TutWritingRMIClients_Client`*
   a. Open the *Java Perspective* and chose the *Package Explorer View*
   b. Open the *src*-folder
   c. Open the *com.example.rmi.clients*-package
   d. Double click on *EjbClientIIOP.java*
   e. Select in the Menue Run/Run As/Java Application

Result

In the Console View the following Output appears in case of a successful execution of the client:



Further Information

- RMI
- RMI-P4 on the SAP Web Application Server
- RMI-IIOP on the SAP Web Application Server
- CORBA

About the author

Johannes Hamel is a technology consultant with wide experience on different J2EE platforms and open source products. He is a member of the Platform Ecosystem – Market Development Engineering team for SAP and is currently focused on the SAP NetWeaver platform.