Interstage Application Server

V7.0

J2EE User's Guide

Trademarks

Trademarks of other companies are used in this user guide only to identify particular products or systems:

Product	Trademark/Registered Trademark
Microsoft, Visual Basic, Visual C++, Windows, Windows NT, Internet Information Server, and Internet Explorer	Registered trademarks of Microsoft Corporation in the U.S.A. and other countries
Sun, Solaris OE, Java, and other trademarks containing Java	Trademarks of Sun Microsystems, Inc., in the U.S.A. and other countries
UNIX	Registered trademark in the U.S.A. and other countries, licensed exclusively through X/Open Company Ltd
Netscape, Netscape FastTrack Server, Netscape Enterprise Server, and Netscape Navigator	Registered trademarks of Netscape Communications Corporation in the U.S.A. and other countries
CORBA, Object Management Group, OMG, OMG IDL, IIOP, Object Request Broker, and ORB	Trademarks or registered trademarks of Object Management Group, Inc., in the U.S.A. and other countries
Interstage and ObjectDirector	Registered trademarks of Fujitsu Limited

This document contains technology relating to strategic products controlled by export control laws of the producing and/ or exporting countries. This document or a portion thereof should not be exported (or re-exported) without authorization from the appropriate government authorities in accordance with such laws.

Fujitsu Limited

First Edition (March 2005)

Preface

Purpose of this Document

This manual is the Interstage Application Server J2EE User's Guide.

This manual explains the general outlines of J2EE, environment construction for J2EE components, and operations of J2EE applications that are needed to develop and operate applications using J2EE components of Interstage.

This manual is intended for the following readers:

- Person who develops applications using J2EE components
- Person who operates applications using J2EE components

The functions offered change with the platform or Interstage Application Server product.

The following table provides a list of differences in function.

Operating System	Product	Java Transaction Service (JTS)	Java Message Service (JMS)	J2EE Connector Architecture (Connector)	Enterprise JavaBeans (EJB)	InfoProvider Pro (IPP)	Interstage HTTP Server	HTTP Tunneling	Cluster Service
NT	Plus	-	-	O	O	O	O	-	-
	SE	-	-	O	O	O	O	O	O
	EE	O	O	O	O	O	O	O	O
	Web-J	-	-	-	-	O	O	-	O
Solaris OE	Plus	-	-	O	O	O	O	-	-
	SE	-	-	O	O	O	O	O	O
	EE	O	O	O	O	O	O	O	O

Operating System	Product	Java Transaction Service (JTS)	Java Message Service (JMS)	J2EE Connector Architecture (Connector)	Enterprise JavaBeans (EJB)	InfoProvider Pro (IPP)	Interstage HTTP Server	HTTP Tunneling	Cluster Service
	Web-J	-	-	-	-	O	O	-	O
Linux	Plus	-	-	O	O	-	O	-	-
	SE	-	-	O	O	-	O	O	-
	EE	O	O	O	O	-	O	O	-
	Web-J	-	-	-	-	-	O	-	-

*1) This is realized by the remote linkage with InfoDirectory of NT or Solaris.OE

Who Should Read this Document?

It is assumed that readers of this document have some knowledge of the following:

- Basic knowledge about Windows NT® or Windows® 2000
- Basic knowledge about Java
- Basic knowledge about J2EE
- Basic knowledge about the Internet
- Basic knowledge about the relational database
- Basic knowledge about CORBA
- Basic knowledge about the transaction model (client/server model)

Solaris OE

- Basic knowledge about UNIX

Linux

- Basic knowledge about Linux

For information related to this manual, refer to the following document.

Designing Enterprise Applications with the Java™ 2 Platform, Enterprise Edition - J2EE™ Blueprints

(You can download this document also from the homepage of Sun Microsystems)

Organization of this Document

This manual is organized as shown below:

Part I J2EE Common Edition

- *Chapter 1 Design of J2EE Application*

  This chapter describes the sequence of development of J2EE applications.

- *Chapter 2 Operating J2EE Applications*

  This chapter explains how to install and operate J2EE applications.

- *Chapter 3 JNDI*

  This chapter explains the general outlines of JNDI.

- *Chapter 4 The J2EE Application Security Function*

  This chapter explains the general outlines of the security function and setting method

Part II Servlet/JSP Edition

- *Chapter 5 Functions of the Servlet Service*

  This chapter describes the functions of the Servlet Service.

- *Chapter 6 Web Application Development*

  This chapter explains how to develop Web applications.

- *Chapter 7 How to Call Web Applications*

  This chapter describes how to call Web applications.

Part III EJB Edition

- *Chapter 8 Basic Functions of the EJB ServiceThis chapter explains the basic functions required to use the EJB Service*

- *Chapter 9 EJB Application Development*

  This chapter explains how to develop and test EJB applications and client applications.

- *Chapter 10 How to create Entity Beans*

  This chapter explains how to Entity Beans

- *Chapter 11 How to call EJB Applications*

  This chapter explains how to call EJB applications

- *Chapter 12 DB Access Environment Definition*

  This chapter explains addition/change/deletion of the data source definition by the DB access environment definition

- *Chapter 13 How to Customize Using by EJB Service Operation Command*

  This chapter explains how to customize the EJB application

- *Chapter 14 Using the Interstage JDBC Driver*

  This chapter explains Interstage JDBC Driver used when connecting with SQL Server from EJB application.

Part IV JTS/JTA Edition

- *Chapter 15 Using Java Transaction API (JTA)*

  This chapter explains using the functions provided in the database service in applications.

Part V JMS Edition

- *Chapter 16 Environment Settings for Interstage JMS*

  This chapter explains the environment settings entered prior to using the Interstage JMS.

- *Chapter 17 Developing a JMS Application*

  This chapter explains developing a JMS application.

Part VI Connector Edition

- *Chapter 18 Basic Functions of the Interstage Connector*

  The fundamental function of Connector is explained.

Part VII Tool Edition

- *Chapter 19 J2EE Resource Access Definition*

  This chapter describes the creation and editing of resource access definitions using J2EE Resource Access Definition.

Part VIII Appendixes

- *Appendix A FJVM*

  This appendix explains the difference between FJVM and the original VM.

- *Appendix B JDK1.3.1 and JDK1.4.2*

  This appendix explains document information on JDK.

Table of Contents

Chapter 7 How to Call Web Applications

Chapter 8 Basic Functions of the EJB Service

Chapter 9 EJB Application Development

Part I

J2EE Common Edition

Chapter 1

Design of J2EE Application

Scope of J2EE Specifications to be Supported

The J2EE platform is a standard environment for executing a J2EE application.

The J2EE platform provided by Interstage consists of the following elements.

- J2EE deployment specification

  A standard for defining the common packaging method for the applications to be deployed on the J2EE-compatible platform

- Java technology standard for the J2EE platform

  A set of standards that all J2EE platforms must support

- IETF standard for the J2EE platform

  A set of standards that all J2EE platform products must support and that is defined by IETF (Internet Engineering Task Force)

- CORBA standard for J2EE platform

  A set of CORBA standards that the J2EE platform conforms to in order to maintain middle layer interoperability

The J2EE platform provided by Interstage defines various functions required to implement company-scale multi-layer services. It conforms to verified open standards to maximize its employment scalability and transportability.

Functions to be Provided as J2EE Components

A J2EE component indicates an application-level software unit that conforms to the J2EE architecture. It provides Interstage with components that conform to the J2EE version 1.3 rules and assumes that the operation environment is JDK1.3 or later. Note that the Interstage functions provided in the J2EE version 1.2 rules are included in the J2EE version 1.3 rules and guaranteed in the J2EE 1.3 plus JDK1.3 environment.

Interstage provides the following services.

- Servlet service (JServlet)

  The Servlet service is a component that controls Web application execution on the Web server.

- EJB service (Interstage EJB)

  The EJB service is a component for executing server applications that conform to the EJB2.0 rules.

- JNDI

  JNDI does not define resource information to be used in each application, but rather provides a JNDI service provider function that can be used in all applications that operate in the Interstage environment.

- JDBC

  JDBC provides the database-independent API used by Java applications to access databases. Interstage provides functions to work with a JDBC driver provided by each database. For details, refer to 'Environment Setup when JDBC (Database) is Referenced' in Chapter 3.

- Java Transaction Service (JTS)

  JTS is a component that provides a service that eliminates the need for being conscious of specific implementation when an application accesses the transaction.

- Java Message Service (JMS)

  JMS is a component that provides asynchronous communication that is implemented on the basis of the JMS rules and is reliable in the distributed environment.

- J2EE Connector Architecture (connector)

  The connector is a component for connecting to an ERP system and main frame located in the EIS layer and a corporate information system such as a database.

- JavaMail

  JavaMail is a component that provides APIs making it possible by the use of Java to implement mail sending and receiving functions independent of environments and protocols and to create applications. Version 1.2 of JavaMail is included in this package. SMTP, IMAP, and POP3 are also included as providers.

Environment Where J2EE Applications are Operated (IJServer)

Interstage Application Server uses a concept called Interstage JavaTM Server (from now on, referred to as IJServer) to enhance operability this has a J2EE application operation environment.

What is IJServer

IJServer is a logical concept that includes EJB and Servlet containers and J2EE application execution environments. It is located at the upper part of these containers.

IJServer operates on the application operation function called 'WorkUnit', an Interstage Application Server feature. By operating as the IJServer WorkUnit, IJServer can use sophisticated application operations/monitoring functions provided by the WorkUnit. IJServer is created using the Interstage Management Console.

Refer to the 'Interstage Operator's Guide' for details of the WorkUnit function and Interstage Management Console.

IJServer Types

IJServer is classified into four types that can be selected according to the purpose.

Normally, the default type, 'Contains Web Applications and EJB Applications (run in single Java VM)', is used.

The IJServer types are shown below.

- Contains Web Applications and EJB Applications (run in single Java VM)

  Web and EJB applications can be run on a single JavaVM. This way, EJB can be invoked quickly from Servlet/JSP, and the memory resources can be saved since applications operate on the same JavaVM.

  It is also possible to operate only Web applications.

- Contains Web Applications and EJB Applications (run in separate Java VMs)

  Web applications and EJB applications can be operated on separate JavaVMs. Although memory resources are used up by operating the applications on separate JavaVMs, a process concurrency can be set for each JavaVM, and a process failure risk can be distributed.

- Contains Web Applications only

  The JavaVM can be used with Web applications only.

- Contains EJB Applications only

  The JavaVM can be used with EJB applications only.

The outline for each type is shown below.

Notes

- When Web and EJB applications are operated on the same JavaVM, EJB applications cannot be invoked from another IJServer application (or EJB client application).
- When Web and EJB applications are operated on the same JavaVM, it is not possible to deploy the EJB applications only.
- When only the EJB applications are deployed on the IJServer where both Web and EJB applications are operated on separate JavaVMs, only the EJB JavaVM is started.
- When only the Web applications are deployed on the IJServer where both Web and EJB applications are operated on separate JavaVMs, only the Web JavaVM is started.
- When EJB applications are to be deployed on the IJServer where only Web applications are operated, the EJB application deployment processing is skipped. When Web applications are to be deployed on the IJServer where only EJB applications are operated, the Web application deployment processing is skipped. Therefore, if an EAR file contains both Web and EJB applications, the environment can be configured by deploying one EAR file on the IJServer where only Web applications are operated and another one on the IJServer where only EJB applications are operated. This enables the user to operate both Web and EJB applications on separate JavaVMs.

IJServer File Configuration

IJServers created and applications deployed by the Interstage Management Console are managed as follows.

Windows

J2EE common directory\ijserver\

(The default J2EE common directory is C:\Interstage\J2EE\var\deployment.)

Solaris OE Linux

/opt/FJSVj2ee/var/deployment/ijserver/

<table>
<tr><td colspan="4">Upper directory <- -> lower directory</td></tr>
<tr><td rowspan="7">IJServer name</td><td>apps (*1)</td><td colspan="2">Module name</td></tr>
<tr><td>current (*2)</td><td>IJServer name</td><td>PID</td></tr>
<tr><td>distribute (*3)</td><td colspan="2">Module name</td></tr>
<tr><td>log (*4)</td><td colspan="2">Process serial number</td></tr>
<tr><td rowspan="2">Shared (*5)</td><td colspan="2">classes</td></tr>
<tr><td colspan="2">lib</td></tr>
<tr><td colspan="3">work (*6)</td></tr>
<tr><td colspan="3">ext (*7)</td></tr>
</table>

*1) Files of the deployed application are extracted in the directory with the name of the module. If an EAR file is deployed, the WAR/ejb-jar/RAR file in it is also extracted. In addition, if the EAR file to be deployed contains the Shared directory, jar files and class files in this directory are files that are commonly used by applications in EAR. Store jar files in the Shared/lib directory, and store class files in the Shared/classes directory. Also, the Shared directory is a load target even if it is created after the deployment.

*2) JavaVM process current directory. The latest is the one with the IJServer name. Up to five generations are backed up with the names IJServer name.old1, IJServer name.old2,…,IJServer name.old5 when the work unit is activated. The PID directories under the IJServer name are current directories for JavaVM process containers of the corresponding PIDs. Note that the current directory location can be changed by selecting [WorkUnit] > [IJServer Name] > [Environment Settings] and then selecting 'Set WorkUnit' on the Interstage Management Console.

*3) A directory with the module name is created, and in this directory, the client distribution data of EJB and the CORBA/SOAP server gateway file are stored. When an EAR file is deployed, a directory with the name of the ejb-jar file is created in the directory with the same name as the module name, and the file is stored in this directory.

*4) JavaVM process log directory. This includes the same number of process serial number directories as for the process concurrency of the IJServer. A log file is output to this directory. Note that the log directory location can be changed by selecting [WorkUnit] > [IJServer Name] > [Environment Settings] and then selecting 'Set WorkUnit' on the Interstage Management Console. It can also be referenced by clicking [WorkUnit] > 'WorkUnit Name', and then by clicking the [Log Reference] tab.

*5) jar files and class files commonly used by applications in IJServer are stored. Store jar files in the Shared/lib directory, and class files in the Shared/classes directory.

Class files and jar files in the Shared directory are not targets of HotDeploy or the class auto reload. When class files and jar files in the Shared directory are overwritten and deleted, they are not enabled until IJServer is rebooted.

Windows

jar files in the Shared/lib directory cannot be overwritten and deleted when IJServer is operated.

*6) Stores container temporary files (JSP compiler results).

*7) Store a JDBC driver in lib of Shared instead of an ext directory.

Current Directory of IJServer

By default, the current directory of IJServer is the following directory:

Windows

J2EE common directory\ijserver\[IJServer-name]\current\[IJServer-name *]\[Process-ID]

(The default J2EE common directory is C:\Interstage\J2EE\var\deployment.)

Solaris OE Linux

/opt/FJSVj2ee/var/deployment/ijserver/[IJServer-name]/current/[IJServer-name *]/[Process-ID]

Backup copies of [IJServer-name *] are created at the start of the WorkUnit and they are named IJServer-name.old1, IJServer-name.old2, ... , IJServer-name.old5. The name of the latest directory is IJServer-name.

Each underlined part above can be changed by selecting [WorkUnit] > [IJServer Name] > [Environment Settings] tab, then using 'WorkUnit configuration' of the Interstage Management Console.

Also, if 'Unique current directory in IJServer' is checked at the specification, the specified directory itself becomes the current directory, and the current directory actually used can be made unique in IJServer.

The following table indicates the relation between the current directory and the settings of 'WorkUnit configuration' that are made by selecting [WorkUnit] > [IJServer Name] > [Environment Settings] tab from the Interstage Management Console:

Option of 'current directory'	**Directory Name**	**'Unique current directory in IJServer'**	**Current Directory**
Default directory structure	(Specification is impossible.) Windows C:\Interstage\J2EE\var\deployment\ijserver (When J2EE common directory is the default directory) Solaris OE Linux /opt/FJSVj2ee/var/deployment/ijserver	(Invalid)	Windows C:\Interstage\J2EE\var\deployment\ijserver\[IJServer-name]\current\[IJServer-name *]\[Process-ID] Solaris OE Linux /opt/FJSVj2ee/var/deployment/ijserver/[IJServer-name]/current/[IJServer-name *]/[Process-ID]
Specification by user	Specification example: Windows C:\tmp\current Solaris OE Linux /tmp/current	When checked	Windows C:\tmp\current Solaris OE Linux /tmp/current
		When not checked	Windows C:\tmp\current\[IJServer-name]\current\[IJServer-name *]\[Process-ID] Solaris OE Linux /tmp/current/[IJServer-name]/current/[IJServer-name *]/[Process-ID]

Note

If 'Unique current directory in IJServer' is checked, pay attention to the following points:

- The current directory is not generation-managed.
- If the specified directory does not exist, it is created anew.
- The specified directory is not deleted even when IJServer is deleted or the current directory is updated.

Solaris OE Linux

- In the cases shown below, there is a possibility that core files are overwritten, and information for investigation may not be obtained when a problem occurs. Therefore, it is recommended that 'Unique current directory in IJServer' not be checked.
  - For the concurrency number of processes of the WorkUnit, two or more is specified.
  - The WorkUnit is restarted.

Class Used by IJServer

IJServer automatically loads the classes described in Class List used by IJServer. The priorities that the class is loaded to depend on the setup of the class, separation of class loaders, and the search order of class loaders. Refer to Class Loader for details.

If there is a class that the user wants to use prior to the class required to operate the container, copy the jar file to the ext directory. In this case, note the following.

- If there are multiple jar files in the ext directory, those files are set in the class path in an undefined order.
- To load the jar files in the ext directory, reboot the IJServer.
- If a jar file in the ext directory and the container have the same class, the container also operates using that class. Because this may cause the container to malfunction, verify the operation fully before using this function.

Also, it can be used for the IJServer class by including a class in the J2EE application, giving priority. In this case, since IJServer does not use the class in the J2EE application even if a class of a same name exists, you are recommended to include a class in the J2EE application. Refer to Class Settings used by IJServer for details.

When the IJServer is started up, the class path information at startup is output to the starting information (info.log) and the container log (container.log). By viewing the starting information and the container log, the class path settings can be checked. The starting information and the container log can be referenced by clicking [WorkUnit] > 'WorkUnit Name', and then by clicking the [Log Reference] tab.

Windows

starting information : IJServer name\log\[Process serial number]\info.log

container log : IJServer name\log\[Process serial number]\container.log

Solaris OE Linux

starting information : IJServer name/log/[Process serial number]/info.log

container log : IJServer name/log/[Process serial number]/container.log

Class List used by IJServer

Windows

No.	Class File		Class Loader
1	Class path required to operate containers (class path when Interstage is installed in 'C\Interstage' is indicated.)	C:\Interstage\J2EE\lib\ijserverboot.jar C:\Interstage\J2EE\lib\jsse.jar C:\Interstage\J2EE\lib\jcert.jar C:\Interstage\J2EE\lib\jnet.jar C:\Interstage\J2EE\lib\isj2ee7.jar C:\Interstage\odwin\etc\class\ODjava2.jar(*1)(*8) C:\Interstage\odwin\etc\class\ODjava4.jar(*2)(*8) C:\Interstage\ots\lib\fjtscorba13.jar(*1) C:\Interstage\ots\lib\fjtscorba14.jar(*2)	System class loader
2	The XML parser specified by 'xml_parser' in environment setup of WorkUnit	C:\Interstage\J2EE\lib\crimson.jar(*3) C:\Interstage\J2EE\lib\jaxp.jar(*3) System drive:\Program Files\Common Files\FujitsuXML\xmlpro.jar(*4) The jar file in the directory specified by 'other directories'(*5)	System class loader
3	Class path required to operate containers (class path when Interstage is installed in 'C\Interstage' is indicated.)	C:\Interstage\J2EE\lib\isj2eert.jar C:\Interstage\J2EE\lib\isjaxp.jar C:\Interstage\lib\isadmin_scs.jar C:\Interstage\jms\lib\fjmsprovider.jar C:\Interstage\eswin\lib\esnotifyjava2.jar(*1) C:\Interstage\eswin\lib\esnotifyjava4.jar(*2) C:\Interstage\ots\lib\fjtsserverimpl13.jar(*1) C:\Interstage\ots\lib\fjtsserverimpl14.jar(*2) C:\Interstage\ejbcl\lib\fjcontainer72.jar(*1)(*6) C:\Interstage\ejb\lib\fjcontainer72.jar(*1)(*7) C:\Interstage\ejbcl\lib\fjcontainer74.jar(*2)(*6) C:\Interstage\ejb\lib\fjcontainer74.jar(*2)(*7) C:\Interstage\F3FMsoap\lib\issoap.jar(*1) C:\Interstage\F3FMsoap\lib\issoap4.jar(*2) %JAVA_HOME%\lib\fmoni.jar(*9) %JAVA_HOME%\jre\lib\fmoni.jar(*9) C:\Interstage\F3FMsoap\lib\issoapsec.jar C:\Interstage\F3FMuddic\lib\fjuddi4.jar C:\Interstage\F3FMuddic\lib\isplugin.jar C:\Interstage\J2EE\lib\fjjca1_0.jar C:\Interstage\J2EE\lib\providerutil.jar C:\Interstage\J2EE\lib\fscontext.jar %JAVA_HOME%\lib\tools.jar(*9) C:\Interstage\J2EE\lib\ijserverw.jar C:\Interstage\ejb\lib\fjcdomdef2.jar C:\Interstage\lib\isjmxserver.jar	Interstage class loader

*1 Set when JDK1.3/JRE1.3 system is used.

*2 Set when JDK1.4/JRE1.4 system is used.

*3 Set when 'xml_parser' in environment setup of WorkUnit is specified with 'Crimson'.

*4 Set when 'xml_parser' in environment setup of WorkUnit is specified with 'Fujitsu XML Processor'.

*5 Set when 'xml_parser' in environment setup of WorkUnit is specified with 'Others'.

*6 For Web-J Edition

*7 For other than Web-J Edition

*8 Because the preinstalled Java library of the CORBA service is automatically set, this is not loaded even if the library for Portable-ORB is set to the class path for WorkUnit setup. When the CORBA service environment definition needs to be changed, change the preinstalled type setting.

*9 %Java_HOME% is the JRE/JDK install directory defined in the \Interstage\J2EE\etc\java_config.txt file.

Solaris OE

No.	Class File		Class Loader
1	Class path required to operate containers	/opt/FJSVj2ee/lib/ijserverboot.jar /opt/FJSVj2ee/lib/jsse.jar /opt/FJSVj2ee/lib/jcert.jar /opt/FJSVj2ee/lib/jnet.jar /opt/FJSVj2ee/lib/isj2ee7.jar /opt/FSUNod/etc/class/ODjava2.jar(*1)(*8) /opt/FSUNod/etc/class/ODjava4.jar(*2)(*8) /opt/FSUNots/lib/fjtscorba13.jar(*1) /opt/FSUNots/lib/fjtscorba14.jar(*2)	System class loader
2	The XML parser specified by 'xml_parser' in environment setup of WorkUnit	/opt/FJSVj2ee/lib/crimson.jar(*3) /opt/FJSVj2ee/lib/jaxp.jar(*3) /opt/FJSVxmlpc/lib/xmlpro.jar(*4) The jar file in the directory specified by 'other directories'(*5)	System class loader

No.	Class File		Class Loader
3	Class path required to operate containers	/opt/FJSVj2ee/lib/isj2eert.jar /opt/FJSVj2ee/lib/isjaxp.jar /opt/FJSVisscs/lib/isadmin_scs.jar /opt/FJSVjms/lib/fjmsprovider.jar /opt/FJSVes/lib/esnotifyjava2.jar(*1) /opt/FJSVes/lib/esnotifyjava4.jar(*2) /opt/FSUNots/lib/fjtsserverimpl13.jar(*1) /opt/FSUNots/lib/fjtsserverimpl14.jar(*2) /opt/FJSVejbcl/lib/fjcontainer72.jar(*1)(*6) /opt/FJSVejbcl/lib/fjcontainer74.jar(*2)(*6) /opt/FJSVejb/lib/fjcontainer72.jar(*1)(*7) /opt/FJSVejb/lib/fjcontainer74.jar(*2)(*7) /opt/FJSVsoap/lib/issoap.jar(*1) /opt/FJSVsoap/lib/issoap4.jar(*2) $JAVA_HOME/lib/fmoni.jar(*9) $JAVA_HOME/jre/lib/fmoni.jar(*9) /opt/FJSVsoap/lib/issoapsec.jar /opt/FJSVuddic/libfjuddi4.jar /opt/FJSVuddic/lib/isplugin.jar /opt/FJSVj2ee/lib/fjjca1_0.jar /opt/FJSVj2ee/lib/providerutil.jar /opt/FJSVj2ee/lib/fscontext.jar $JAVA_HOME/lib/tools.jar(*9) /opt/FJSVj2ee/lib/ijserverw.jar /opt/FJSVejb/lib/fjcdomdef2.jar /opt/FJSVisjmx/lib/isjmxserver.jar	Interstage class loader

*1 Set when JDK1.3/JRE1.3 system is used.

*2 Set when JDK1.4/JRE1.4 system is used.

*3 Set when 'xml_parser' in environment setup of WorkUnit is specified with 'Crimson'.

*4 Set when 'xml_parser' in environment setup of WorkUnit is specified with 'Fujitsu XML Processor'.

*5 Set when 'xml_parser' in environment setup of WorkUnit is specified with 'Others'.

*6 For Web-J Edition

*7 For other than Web-J Edition

*8 Because the preinstalled Java library of the CORBA service is automatically set, this is not loaded even if the library for Portable-ORB is set to the class path for WorkUnit setup. When the CORBA service environment definition needs to be changed, change the preinstalled type setting.

*9 %Java_HOME% is the JRE/JDK install directory defined in the /opt/FJSVj2ee/etc/java_config.txt file.

Linux

No.	Class File		Class Loader
1	Class path required to operate containers	/opt/FJSVj2ee/lib/ijserverboot.jar /opt/FJSVj2ee/lib/jsse.jar /opt/FJSVj2ee/lib/jcert.jar /opt/FJSVj2ee/lib/jnet.jar /opt/FJSVj2ee/lib/isj2ee7.jar /opt/FJSVod/etc/class/ODjava2.jar(*1)(*8) /opt/FJSVod/etc/class/ODjava4.jar(*2)(*8) /opt/FJSVots/lib/fjtscorba13.jar(*1) /opt/FJSVots/lib/fjtscorba14.jar(*2)	System class loader
2	The XML parser specified by 'xml_parser' in environment setup of WorkUnit	/opt/FJSVj2ee/lib/crimson.jar(*3) /opt/FJSVj2ee/lib/jaxp.jar(*3) /opt/FJSVxmlpc/lib/xmlpro.jar(*4) The jar file in the directory specified by 'other directories'(*5)	System class loader
3	Class path required to operate containers	/opt/FJSVj2ee/lib/isj2eert.jar /opt/FJSVj2ee/lib/isjaxp.jar /opt/FJSVisscs/lib/isadmin_scs.jar /opt/FJSVjms/lib/fjmsprovider.jar /opt/FJSVes/lib/esnotifyjava2.jar(*1) /opt/FJSVes/lib/esnotifyjava4.jar(*2) /opt/FJSVots/lib/fjtsserverimpl13.jar(*1) /opt/FJSVots/lib/fjtsserverimpl14.jar(*2) /opt/FJSVejbcl/lib/fjcontainer72.jar(*6)(*1) /opt/FJSVejbcl/lib/fjcontainer74.jar(*6)(*2) /opt/FJSVejb/lib/fjcontainer72.jar(*7)(*1) /opt/FJSVejb/lib/fjcontainer74.jar(*7)(*2) /opt/FJSVsoap/lib/issoap.jar(*1) /opt/FJSVsoap/lib/issoap4.jar(*2) $JAVA_HOME/lib/fmoni.jar(*9) $JAVA_HOME/jre/lib/fmoni.jar(*9) /opt/FJSVsoap/lib/issoapsec.jar /opt/FJSVuddic/libfjuddi4.jar /opt/FJSVuddic/lib/isplugin.jar /opt/FJSVj2ee/lib/fjjca1_0.jar /opt/FJSVj2ee/lib/providerutil.jar /opt/FJSVj2ee/lib/fscontext.jar $JAVA_HOME/lib/tools.jar(*9) /opt/FJSVj2ee/lib/ijserverw.jar /opt/FJSVejb/lib/fjcdomdef2.jar /opt/FJSVisjmx/lib/isjmxserver.jar	Interstage class loader

*1 Set when JDK1.3/JRE1.3 system is used.

*2 Set when JDK1.4/JRE1.4 system is used.

*3 Set when 'xml_parser' in environment setup of WorkUnit is specified with 'Crimson'.

*4 Set when 'xml_parser' in environment setup of WorkUnit is specified with 'Fujitsu XML Processor'.

*5 Set when 'xml_parser' in environment setup of WorkUnit is specified with 'Others'.

*6 For Web-J Edition

*7 For other than Web-J Edition

*8 Because the preinstalled Java library of the CORBA service is automatically set, this is not loaded even if the library for Portable-ORB is set to the class path for WorkUnit setup. When the CORBA service environment definition needs to be changed, change the preinstalled type setting.

*9 $Java_HOME is the JRE/JDK install directory defined in the /opt/FJSVj2ee/etc/java_config.txt file.

Startup/Shutdown Execution Class

When an IJServer is started up and shut down, an optional Java application can be invoked.

The Java application to be invoked when the IJServer is started up is called a startup time execution class. The Java application to be invoked when the IJServer is shut down is called a shutdown time execution class.

In the execution class, implement the initialization and exit processing to be executed only once for each IJServer and JavaVM process. Then, these processes can be used in the following processing.

- Database initialization and exit processes
- Object lookup process by JNDI
- Invocation of EJB application method

The triggers for an execution class to be invoked are as follows:

- The startup time execution class is invoked before a request to the application is received after the IJServer is started up. This class is invoked before EJB activation if at all.
- The shutdown time execution class is invoked immediately after the request to the application is received when the IJServer is shut down. When the IJServer is forcibly shut down, the shutdown time execution class is not invoked.
- More than one execution class can be specified and the invocation order can be determined.
- The user can invoke an execution class in all processes (JavaVM) or invoke it only once for each IJServer.

Invocation Method	Example of Use	Example
In all processes	Processing that must be done in each process	Object lookup processing by JNDI
Once only	Processing that does not need to be executed for each process and that can be shared between processes	Database initialization and exit processing

How to Create an Execution Class

A special class or interface does not need to be prepared to create an execution class.

Create a Java application that meets the following two conditions.

- Executable from the command line with the main method implemented
- Declared as a public class

An example of a simple execution class is indicated below.

```
package test;
public class UserStartup{
  public static void main(String args[]){
    if(args.length == 1){
      System.out.println(args[0]);
    }
    //Define user implementation here.
  }
}
```

How to Register an Execution Class

To register an execution class, make the following two settings on the Interstage Management Console.

1. Execution class setting

   Store execution class setting information.

2. Class path setting

   Set an execution class in the WorkUnit class path.

Notes

- When the IJServer type is for 'Contains Web Applications and EJB Applications (run in separate Java VMs)', there are two class paths, one for the Servlet container and the other for the EJB container. Set the execution class in the class path for the container to be invoked.
- When the IJServer type is for 'Contains Web Applications and EJB Applications (run in separate Java VMs)', select the container that starts the execution class from one of the following three to respond to container-dependent processing.
  - Web container
  - EJB container
  - Web and EJB containers
- When an exception that occurred in the startup time execution class is thrown to the IJServer WorkUnit, whether to continue the IJServer startup or not can be specified.

- When processing is performed from the execution class to an EJB application on another JavaVM, the IJServer where the EJB application is deployed must be started up before the execution class is started.

  When the IJServer type is for 'Contains Web Applications and EJB Applications (run in separate Java VMs)', the Web container is started before the EJB container. Therefore, a process for an EJB application on the same IJServer cannot be performed from the execution class that is set in the Web container.

  An example where the processing cannot be executed and its workaround are listed below.

Example of Impossible Processing	**User Response**
EJB application lookup processing on the same IJServer from the execution class set in the Web container when the IJServer type is 'Contains Web Applications and EJB Applications (run in separate Java VMs)'	Select one of the following: - Specify the EJB container as the container that starts the execution class. - Deploy and run the EJB application beforehand on a different IJServer from the one where the execution class is set.

- When 'Separation of class loaders' is 'Separate between EARs' or 'Separate all', EJB application cannot be called from a startup time execution class or a shutdown time execution class.

Class Loader

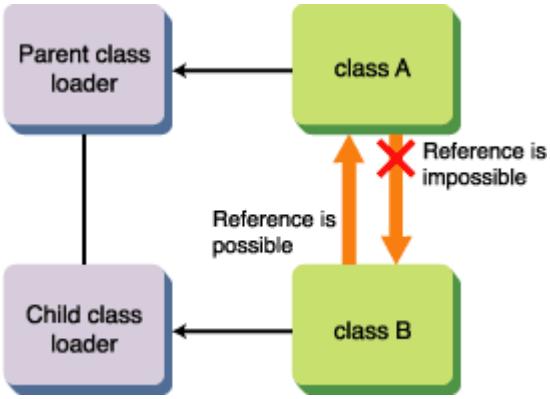
Class loaders of Java provide a function to search for a class file and load classes on the memory.

When developing J2EE application, understand what class loader will load the classes before the development.

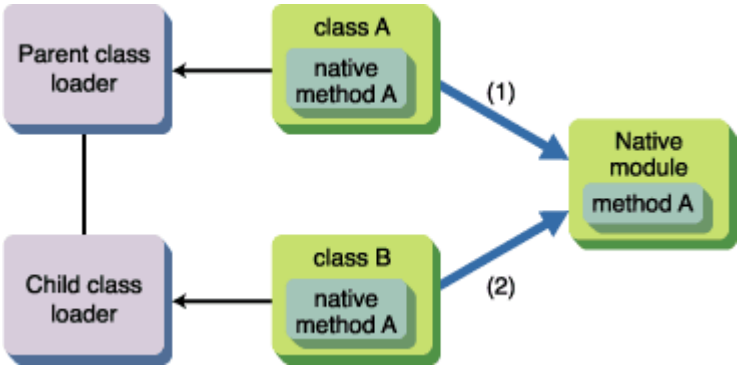
Layer of a Class Loader

Java class loaders have a hierarchical structure consisting of parent class loaders and child class loaders. The relation between a parent loader and child loader resembles that of a super class and subclass of an object.

A class that is loaded by a child class loader can refer to a class that is loaded by the parent class loader, but a class that is loaded by a parent class loader cannot refer to a class loaded by a child class loader.



If Java Native Interface (JNI) is used, further care needs to be taken. When JNI is used, a class loader loads a native module. For Java class loaders, the same native module can be used only from the same class loader.



For (1) and (2), the same native module is supposed to be used, but it can be used only from the side where the native module is first loaded.

If class B works first, 'java.lang.UnsatisfiedLinkError' is thrown to class A.

The above also applies to Interstage.

Loading a Class

Generally, a class loader uses a devolution model to load a class. A class loader has a relevant parent class loader. When a class loader is called to load a class, it devolves the loading of this class on its parent class loader before the class loader itself tries to load the class.

The operations of the Interstage class loader and this transfer model are slightly different to boost the independence of classes used by Interstage and classes used by applications. The following section explains the search order of the Interstage class loader.

Structure of a Class Loader

This section explains the hierarchical structure of a class loader of IJServer.

Class loaders used in IJServer have a hierarchical structure consisting of parent class loaders and child class loaders.

This hierarchical structure of class loaders increases the degree of independence between the system and application and between application programs.

Layers of class loaders can be customized. Refer to Separation of Class Loaders for the customization method.

Each class loader loads the resources in the table below. According to the Interstage default settings, a class loader sequentially loads classes from the top of the table. The loading order of classes can be customized. Refer to Changing the Search Order of Class Loaders for the customization method.

When 'Separation of class loaders' is 'Separate between EARs' or 'Separate all'

Class Loader	Resources to be Loaded	Setting Method
System class loader	XML parser classes	XML parser class environment settings for the WorkUnit
Application class loader	Connector classes Note) When a connector is deployed for IJServer	RAR file
	EJB application classes	ejb-jar file
	Classes used by EJB applications that are not EJB application classes	Manifest classpath in a ejb-jar file
	Classes that are commonly used in applications	Stored in the Shared directory of an EAR file
Webapp class loader	Web application classes	WAR file
	Classes used by the Web application that are not Web application classes	Manifest classpath in a WAR file

Class Loader	Resources to be Loaded	Setting Method
Interstage class loader	Interstage classes	Nothing (it cannot set up)
	Common classes in IJServer	Classpath of the WorkUnit
		Stored in the Shared directory of IJServer
	Classes common to multiple IJServers	Classpath of J2EE property

When 'Separation of class loaders' is 'Do not separate'

Class Loader	Resources to be Loaded	Setting Method
System class loader	XML parser classes	XML parser class environment settings for the WorkUnit
	Interstage classes	Nothing (it cannot set up)
	Common classes in IJServer	Classpath of the WorkUnit
		Stored in the Shared directory of IJServer
	Classes common to multiple IJServers	Classpath of J2EE property
		Environment variable: CLASSPATH
	Connector classes	RAR file
	EJB application classes	ejb-jar file
	Classes used by EJB applications that are not EJB application classes	Manifest classpath in a ejb-jar file
	Classes that are commonly used in applications	Stored in the Shared directory of an EAR file
Webapp class loader	Web application classes	WAR file
	Classes used by the Web application that are not Web application classes	Manifest classpath in a WAR file

Refer to 'Class Used by IJServer' for the details of the resources which IJServer loads automatically.

Separation of Class Loaders

According to the default settings of Interstage, class loaders are separated between EARs.

The separation of class loaders influences the referencing relation between applications and activation change of applications.

This section explains the function to customize the separation of class loaders.

Setting Method

The separation method of class loaders can be set up by one of the following methods from the Interstage Management Console:

- [WorkUnit] > [Create] > [Class Loader Environment Settings] > [Class Loader Separation]
- [WorkUnit] > [WorkUnit Name] > [Environment Settings] > [Class Loader Environment Settings] > [Class Loader Separation]

The supposed setting patterns are as shown in the following table:

Setting Value	Setting Pattern
Separate between EARs (the default value)	Used when ejb-jar is deployed separately without making an EAR.
Separate all	Used when deployment is done with making an EAR.
Do not separate	Used when an application is referred to between EARs, or when an EJB application or connector is loaded by the system class loader. V6 compatible mode, which provides the same structure of class loaders as those of V6 This is used when HotDeploy is not used and an application developed with Interstage V6 is migrated (or for an application that does not operate under the condition of 'Separate between EARs' or does not operate under the condition of 'Separate all').

Notes

- Because Web application programs do not refer to classes between each other, regardless of the setting value of 'Separate between EARs,' 'Separate all,' or 'Do not separate,' class loaders between Web application programs are separated in all cases.
- When the IJServer type is specified as 'Operate a Web application and EJB application within the same Java VM,' an EJB application that has been deployed in another IJServer cannot be called by a Web application or EJB application.

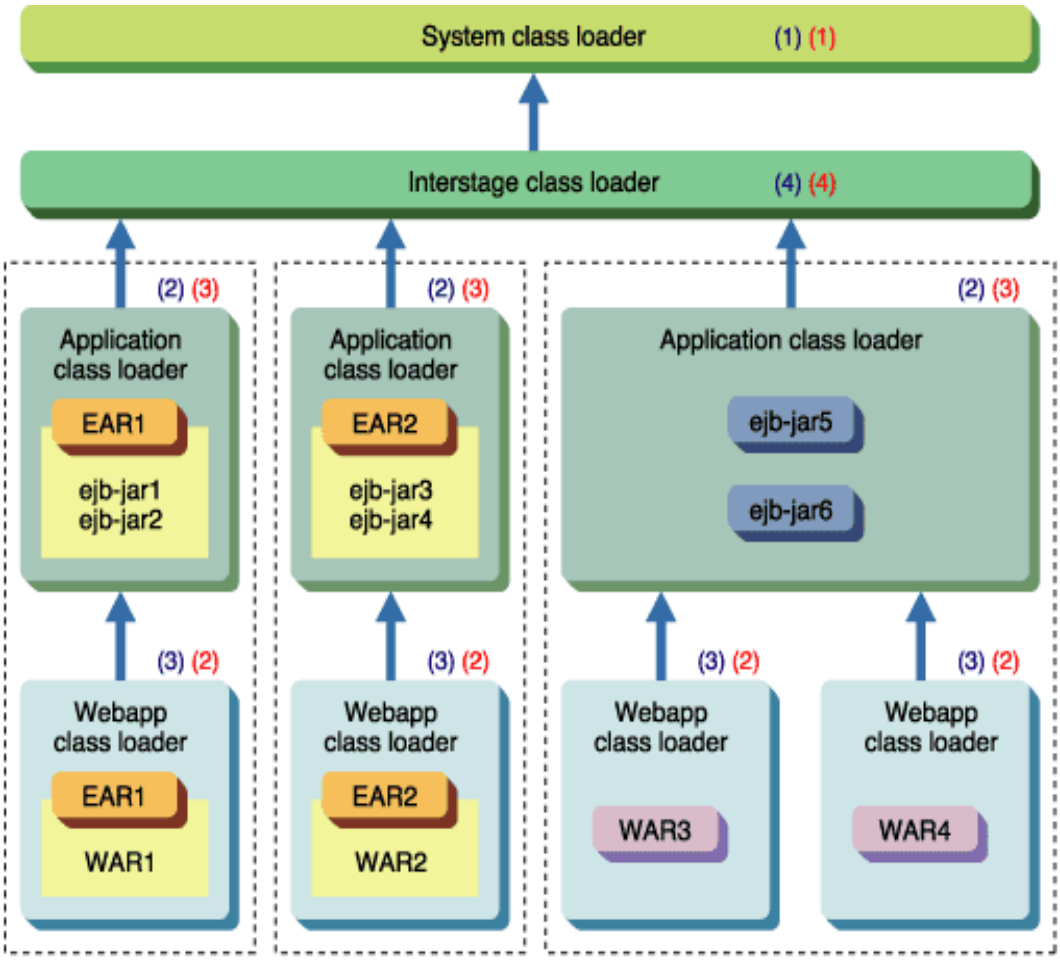
Separation Pattern of Class Loaders

Separate Between EARs

When 'Separate between EARs' is selected, as to EAR deployment, class loaders are separated between EARs as shown in the figure below.

As to ejb-jar, or WAR deployment, class loaders are not separated in units of deployment.

In this case, the activation change of a J2EE application is possible in the units for the parts in dotted lines.

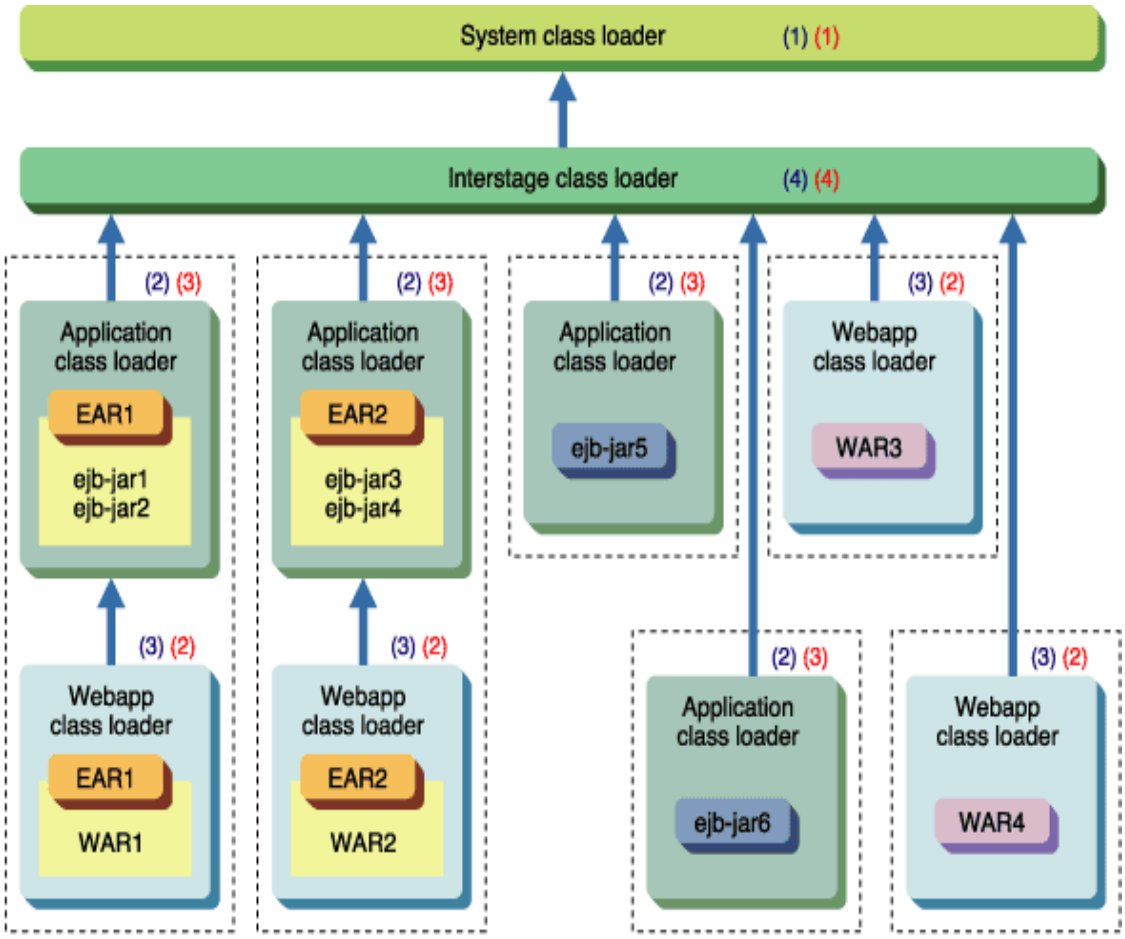


↑ in a figure expresses the child-parent relationship of a class loader. (Terminal points are parents)
(1), (2), (3), and (4) express the search order of the class loader at the time of setting up "Parent is first".
(1), (2), (3), and (4) express the search order of the class loader at the time of setting up "Parent is later ".

Separate All

When 'Separate all' is selected, class loaders are separated in the deployment units as shown in the figure below.

In this case, the activation change of a J2EE application is possible in the units framed by the dotted lines. However, cross-reference between an EJB application of ejb-jar5 and ejb-jar6, and reference from a Web application of WAR3 or WAR4 to an EJB application are impossible. So, different application can use classes that have the same package names and same class names.



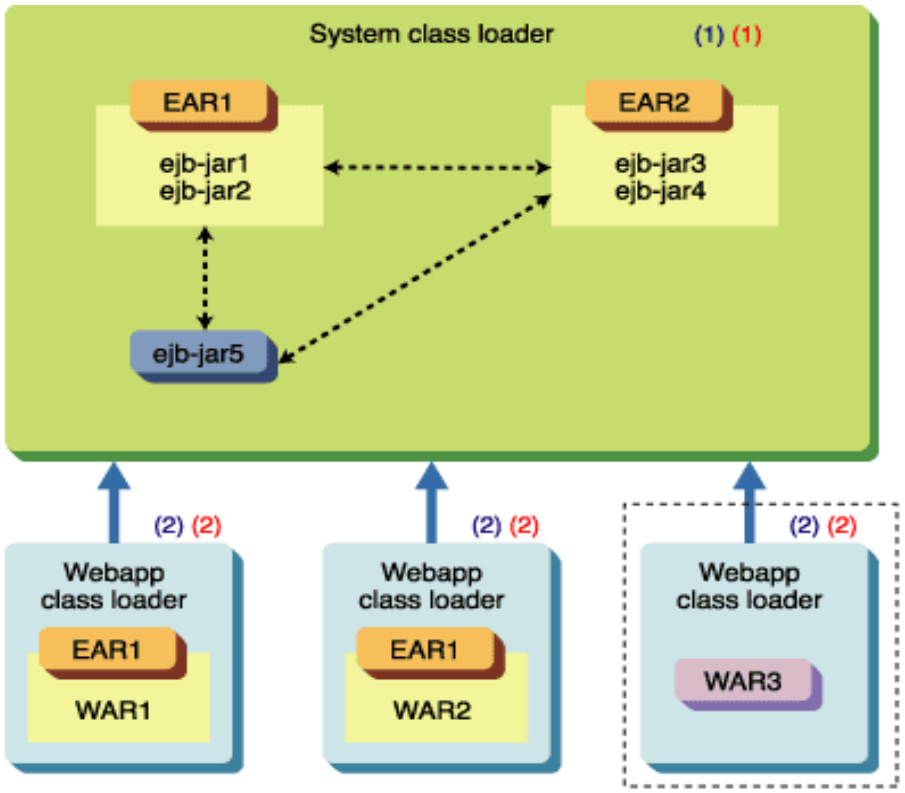
↑ in a figure expresses the child-parent relationship of a class loader. (Terminal points are parents)
(1), (2), (3), and (4) express the search order of the class loader at the time of setting up "Parent is first".
(1), (2), (3), and (4) express the search order of the class loader at the time of setting up "Parent is later ".

Do Not Separate

When 'Do not separate' is selected, all the classes except Web application classes are loaded with the 'system class loader' as shown in the figure below.

In this case, the activation change of a J2EE application is performed only for the Web application that deploys WAR files (the parts framed by the dotted lines in the figure below).

As for the reference between applications, EJB applications can refer to the class of each other. In addition, from Web applications, the classes of all the EJB applications can be referred to.



↑ in a figure expresses the child-parent relationship of a class loader. (Terminal points are parents)
Also, ↕ expresses the reference relation of a class.
(1) and (2) express the search order of the class loader at the time of setting up "Parent is first".
(1) and (2) express the search order of the class loader at the time of setting up "Parent is later ".

Changing the Search Order of Class Loaders

In the default Interstage settings, classes are searched in order of Application class loader first, followed by Webapp class loader.

By making the search order of the class loader 'Parent after', classes can be searched in order of Webapp class loader first, followed by Application class loader. The advantage of making the search order of the class loader 'Parent after' is that, for example, if there are classes in the application with the same class name, making the search order of the class loader 'Parent after' means that the class loader overrides the loaded class with the parent class loader so that it is possible to have independent versions of classes.

If 'Do not Separate' is set for the class loader, there are only two types of class loader, the Webapp class loader and the system class loader. Since the system class loader is always searched for first, it is not necessary to set up the search order of the class loader.

Setting Method

The search order of class loaders can be set up by one of the following methods from the Interstage Management Console:

- [WorkUnit] > [Create] > [Common Definition] > [Searching order of Class Loaders]
- [WorkUnit] > [WorkUnit Name] > [Environment Settings] > [Common Definition] > [Searching order of Class Loaders]

The search order of class loaders changes according to the setting value as shown in the following table:

Setting Value	Search Order
Parent is first (default setting)	System class loader Application class loader Webapp class loader Interstage class loader
Parent is later	System class loader Webapp class loader Application class loader Interstage class loader

Priority Exception

As for the priority of classes, there are the following two exceptions:

- JDK classes

  Because JDK classes are always loaded first, they cannot be replaced by a user.

- Classes whose names begin with particular package names

  The loading of classes whose names begin with any of the package names below is always devolved on their parent class loader.

  If classes whose names begin with any of the package names below is included in a parent class loader, a child class loader cannot replace classes.

Package Name	Type
javax	Java extensions
org.xml.sax	SAX 1 & 2
org.w3c.dom	DOM 1 & 2
com.fujitsu.interstage	Interstage class
com.fujitsu.ObjectDirector	Interstage class (ObjectDirector)

Class Settings used by IJServer

This section explains the setting method of classes used by IJServer and applications.

XML Parser

Refer to 'Settings of XML Parser'.

Classes Common to Multiple IJServers

Classes used commonly to multiple IJServers are specified in the classpath of J2EE property.

For classes common to multiple IJServers, libraries including the JDBC driver, which are used by users in their applications, can be freely specified.

The classpath of J2EE property can be set up with the following method from the Interstage Management Console:

- [Environment Settings] > [J2EE properties] > [Classpath]

Note

When specifying a directory name with the extension '.jar' or '.zip' to be a class path, add a path separator to the end of the class path.

Common Classes in IJServer

The setup methods of the classes that are commonly used in IJServer are explained below.

There are two setup methods of the classes commonly used in IJServer; one is to set up the classpath of the WorkUnit, and the other is to store information in the Shared directory of IJServer.

Method of Setting up the Classpath of the WorkUnit

Specify the absolute path to the directory containing the jar file or class file in the classpath of the WorkUnit.

The classpath of the WorkUnit can be set up by one of the following methods from the Interstage Management Console:

- [WorkUnit] > [Create] > [WorkUnit Configuration] > [Classpath]
- [WorkUnit] > [WorkUnit Name] > [Environment Settings] > [WorkUnit Configuration] > [Classpath]

Note

When specifying a directory name with the extension '.jar' or '.zip' to be a class path, add a path separator to the end of the class path.

Method of Storing Files in the Shared Directory in the IJServer Directory

Use one of the following methods for the setup:

- Store the jar files in the Shared/lib directory of IJServer.
- Store the classes file in the Shared/classes directory of IJServer.

If classes with the same package names and same class names are both in the Shared/lib directory and in the Shared/classes directory, the classes in the Shared/classes directory are loaded.

Since class or jar files in this directory are not reactivated or auto-reloaded, if they are overwritten this operation does not take effect until IJServer is restarted.

Note

In order to save resources to a Shared directory, users must have the appropriate level of authority. An administrator may need to change a general user's authority.

Environment Variable: CLASSPATH

When 'Do not separate' is set for the separation of class loaders, the class set in environment variable: CLASSPATH (or the system environment variable when automatic start is used) is loaded by the 'system class loader'.

When 'Separate between EARs' or 'Separate all' is set for the separation of class loaders, the class set in environment variable: CLASSPATH is not loaded.

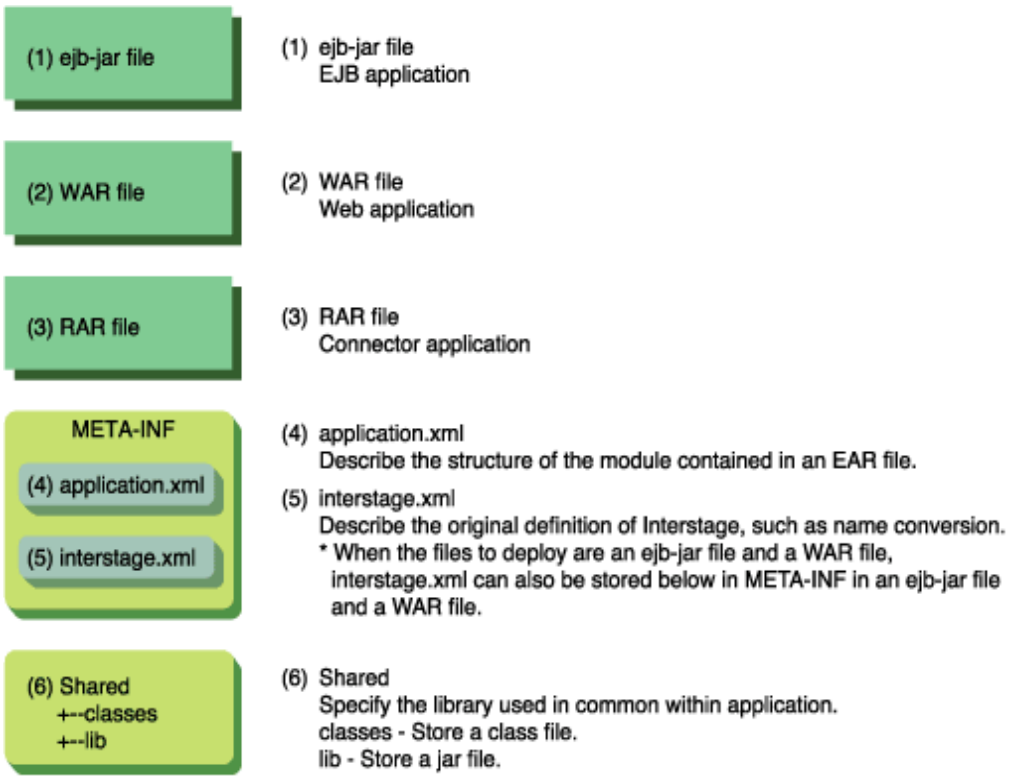
Application Classes

Application classes can be set by deploying EAR file, ejb-jar file, WAR file, and RAR file with the following Interstage Management Console function:

- [WorkUnit] > [WorkUnit Name] > [Deployment] > [Environment Settings] > [Deployment File(s)]

The EAR File Configuration

An EAR file is configured with the following file configuration:



EJB Application Classes

EJB application classes can be set up with one of the following methods:

- Store an ejb-jar file in an EAR file, and deploy it in IJServer.
- Deploy an ejb-jar file in IJServer.

Web Application classes

Web application classes can be set up with one of the following methods:

- Store a WAR file in an EAR file, and deploy it in IJServer.
- Deploy a WAR file in IJServer.

Connector Classes

Connector classes can be set up with one of the following methods:

- Store a RAR file in an EAR file, and deploy it in IJServer.
- Deploy a RAR file in IJServer.
- Deploy a RAR file in the connector service.

The deployment destination of the connector deployed in the connector service must be specified in the classpath of the WorkUnit.

Common Class in an Application

Methods to deploy libraries common to EJB applications and Web applications such as the utility class are explained.

There are the following two methods for using a common class in an application:

- Method by setting up the manifest classpath
- Method by using the Shared directory in an EAR file

Both methods allow commonly used classes such as the utility class to be used from an EJB application and Web application without having them be included in an ejb-jar file or WAR file.

For the above two methods, the ranges where classes can be referred to are different as follows:

Method	Range where Classes can be Referred to
Method by setting up the manifest classpath	Reference is possible only within an EJB application (ejb-jar) and Web application (WAR) to which the manifest classpath is set.
Method by using the Shared directory in an EAR file	Reference is possible from all the classes within an application.

Method of Setting up the Manifest Classpath

Store classes and manifest files that are common within an application in an EAR file as shown below.

1. Store the utility class at the top of the EAR file or in any directory of the EAR file.
2. Describe the following entry in META-INF/MANIFEST.MF in an ejb-jar file or WAR file that uses the utility class:

```
Manifest-Version: 1.0
Class-Path: The jar file containing the utility classes or the directory
containing the class files is specified using the relative path within the
 EAR file.
```

For example, when utility1.jar, utility2.jar, or com.fujitsu.Utility.class is used from web1.war, include in web1.war a manifest file that is defined as follows:

[EAR file configuration]

- web1.war
- utility1.jar
- util/utility2.jar
- util/com/fujitsu/Utility.class

[Manifest file contained in WAR]

```
Manifest-Version: 1.0
Class-Path: utility1.jar util/utility2.jar util
```

Note

If you are creating a manifest file, note the following points:

- A space is required after ':' in the header.
- A description in a line must contain a maximum of 72 bytes in UTF-8 encoding form. If 72 bytes are exceeded, enter a space at the start of the following line, and continue the description after that.

Method of using the Shared Directory in an EAR File

The setup can be made by storing common classes in an application in the Shared directory in the EAR file, and deploying the EAR file in IJServer.

Store a jar file in the Shared/lib directory, and store a class file in the Shared/classes directory.

The Shared directory in an EAR file is the function unique to Interstage, and it is disabled for other application servers.

In addition, the Shared directory is valid only for EAR files, and it is disabled for ejb-jar, RAR, and WAR files.

If classes with the same package names and same class names are both in the Shared/lib directory and in the Shared/classes directory, the classes in the Shared/classes directory are loaded.

The class or jar file in this directory is the target of reactivate and class auto reload.

Settings of XML Parser

This section explains the setup methods of the XML parsers that are used in IJServer.

The following XML parsers can be used for Interstage:

- Crimson (Sun's XML parsers)
- Fujitsu XML processor
- Others

Setting an XML Parser to be used for each IJServer

An XML parser to be used can be set up by one of the following methods from the Interstage Management Console:

- [WorkUnit] > [Create] > [XML Parser Environment Settings] > [XML Parser Used]
- [WorkUnit] > [WorkUnit Name] > [Environment Settings] > [XML Parser Environment Definition] > [XML Parser Used]

Setting values are as follows:

- Crimson (default)
- Fujitsu XML processor

  Note) It is necessary to install Fujitsu XML processor beforehand. If Fujitsu XML processor is not installed, an error occurs.

- Others

  In this case, specify the full path to the directory in which the jar file of the XML parser is stored. Store both an interface such as org.w3c.dom,org.xml.sax, javax.xml.parsers and a implemented class of XML parser into the specified directory. IJServer cannot be started in one of cases.

  Note) A user that starts the WorkUnit must have the read permission to the directory to be specified.

Note

According to the type of the specified XML parser, the XML parser is set up for the start parameter of IJServer as shown below.

XML Parser Type	JDK1.3	JDK1.4
Crimson	Set to '-classpath' at the start of IJServer.	None The XML parser included in JDK is used.
Fujitsu XML processor		Set to '-classpath' at the start of IJServer.
Others		Set to the JavaVM option: '-Djava.endorsed.dir' at the start of IJServer.

Note

An XML parser class cannot be set to below. It is ignored when it sets up.

- J2EE property
- Classpath of the WorkUnit
- Shared directory of IJServer

Specifying an XML Parser to be used for each Application

As with other classes, the XML parser used by each application can be specified with the following settings:

- Include an XML parser in the EAR file.
- Include an XML parser in the ejb-jar file.
- Include an XML parser in the WAR file.

Notes

- The XML parser to be used and the XML parser that has been specified in the XML parser environment settings of IJServer must have interface levels that meet the conditions set out in the points below. If the interface levels are different, the XML parser may not be able to be used. If an XML parser cannot be used, check whether the level of the XML parser' interface used by the application for the XML parser selected by the WorkUnit environment definition is correct. If the level of an interface is not correct, make it the same.

  - JAXP(javax.xml.parsers package)
  - SAX(org.xml.sax package)
  - DOM(org.w3c.dom package)

- If the same XML parser as one that has been specified in the XML parser environment settings of IJServer is specified, the specification is ignored, and the XML parser specified in the XML parser environment settings of IJServer is used.
- When an XML parser type is specified as shown below, it is necessary to specify the XML parser that has been specified in the XML parser environment settings of IJServer.

  If a different XML parser from the one in the XML parser environment settings of IJServer is specified, the start of IJServer fails.

  - Specification with the system property

    DOM : javax.xml.parsers.DocumentBuilderFactory

    SAX : javax.xml.parsers.SAXParserFactory

  Specification by jaxp.properties of the following

  Windows

  When JDK 1.4 is used : C:\Interstage\JDK14\jre\lib\jaxp.properties
  When JRE 1.4 is used : C:\Interstage\JRE14\lib\jaxp.properties
  When JDK 1.3 is used : C:\Interstage\JDK13\jre\lib\jaxp.properties
  When JRE 1.3 is used : C:\Interstage\JRE13\lib\jaxp.properties

  Solaris OE Linux

  When JDK 1.4 is used : /opt/FJSVawjbk/jdk14/jre/lib/jaxp.properties
  When JRE 1.4 is used : /opt/FJSVawjbk/jre14/lib/jaxp.properties
  When JDK 1.3 is used : /opt/FJSVawjbk/jdk13/jre/lib/jaxp.properties
  When JRE 1.3 is used : /opt/FJSVawjbk/jre13/lib/jaxp.properties

Problem Investigation with the Trace Function

When a J2EE application is developed, it is necessary to consider what class loader will load each class.

If an incorrect class loader is used to load a class, the J2EE application may not operate correctly. In order that the investigation of such problems can be facilitated, the trace function is provided.

When a class is loaded, the trace function outputs the information of the class loader that loads the class. The output is recorded in the 'container log'. From the 'container log,' the following can be learned:

- The order in which classes are loaded
- The class loader that loads the class

Output Format

The following are the formats of trace information:

- When the class loader is the 'Webapp class loader' or 'Application class loader'

```
[Time stamp] [Loaded class-name from repository by class-loader-type]
```

- When the class loader is the 'System class loader' or 'Interstage class loader'

```
[Time stamp] [Loaded class-name by class-loader-type]
```

Output Items

<table>
<tr><th>Item</th><th colspan="2">Description</th></tr>
<tr><td>Time stamp</td><td colspan="2">Date and hour of the loading of the class</td></tr>
<tr><td>Class name</td><td colspan="2">Class name of the loaded class (including the package name)</td></tr>
<tr><td>Repository</td><td colspan="2">Storage directory name or jar file name of the loaded class</td></tr>
<tr><td rowspan="6">Class loader type</td><td colspan="2">Name of the class loader that loaded the class</td></tr>
<tr><td>Class Loader</td><td>Displayed Name</td></tr>
<tr><td>System class loader</td><td>System</td></tr>
<tr><td>Interstage class loader</td><td>Interstage</td></tr>
<tr><td>Catalina class loader
Note) Used in the Interstage system</td><td>Catalina</td></tr>
<tr><td>Application class loader</td><td>Application</td></tr>
<tr><td>Webapp class loader</td><td>Webapp</td></tr>
</table>

Notes

- The class trace used by classes that are loaded in the system class loader is not output. For details of the classes that are loaded in the system class loader, refer to Structure of a Class Loader

  To output the class, specify '-verbose:class' in the WorkUnit JavaVM option and then collect the JavaVM trace information.

- The trace of a class loaded in the system class loader is output in the system class loader when the request to load the class is executed. At this time, there is a possibility that the system class loader may collect an already loaded class from the cache and use it.

Log Output Example

```
[24/02/2004 16:17:22:093 +0900] [Loaded com.xxx.ClassA by Interstage]
[24/02/2004 16:17:22:101 +0900] [Loaded com.xxx.ClassB from
c:\Interstage\J2EE\lib\xxx.jar by Webapp]
```

Setting Method

The trace function can be set up by one of the following methods from the Interstage Management Console:

- [WorkUnit] > [Create] > [Common Definition] > [Output Class Loader Trace Information]
- [WorkUnit] > [WorkUnit Name] > [Environment Settings] > [Common Definition] > [Output Class Loader Trace Information]

Setting values are as follows:

- Do not output (default setting)
- Output

Note

The trace function is intended to be used for the debugging during development. It is recommended that this function not be used in the operating environment.

Example of Use

This section describes examples of using the class loader trace function.

[Momentum]

Collect the class loader trace information if the problem described below occurs. The class loader trace function is helpful for resolving this kind of problem.

- The EJB application call from Servlet failed. The container log analysis result detects that ClassCastException has occurred in ClassA.

Analysis Procedure

Execute the analysis according to the following procedure.

1. Click [WorkUnit] > 'WorkUnit Name'. Click the [Environment Settings] tab, and then click [Common Definition] > [Output Class Loader Trace Information]. Select the [Output] checkbox.
2. Execute a reproduction test.
3. Search for the trace information that is output in the container log using ClassA. This detects the following 2 lines.

```
[Loaded ClassA
C:\Interstage\J2EE\var\deployment\ijserver\kaz001\apps\j2eesample.ear\
CartBean.jar by Application]
[Loaded ClassA
C:\Interstage\J2EE\var\deployment\ijserver\kaz001\apps\j2eesample.ear\
j2eesample.war\WEB-INF\classes by Webapp]
```

From the above data, it is judged that ClassA is stored in 2 locations, the EJB application and the Web application.

Investigation

Since the EJB application call from Servlet failed because of the class batting, investigate whether the following support exists.

1. Is it possible to change the application configuration so that the same class does not exist in both the EJB application and Web applications?
2. Is the problem avoidable by making the search order for the class loader 'Parent first'?
3. Is the problem avoidable by making the class loader type 'Do not Separate'?

Notes to be taken when Class Loaders are used

This section explains the notes to be taken when class loaders are used.

Notes to be taken when the JDBC Driver is used

The JDBC driver needs to be loaded by the 'Interstage class loader'. Therefore, define the JDBC driver in the following locations:

- Classpath of J2EE property
- Classpath of the WorkUnit
- Shared directory of IJServer

If a definition contains an error, an error and exception (Exception) occur and the JDBC driver cannot be used.

Notes about using the Connector

When the connector is used, the RAR file may contain a shared library.

In that case, set the deployment directory of the RAR file to the WorkUnit environment setting.

From the Interstage Management Console, set the deployment directory of the RAR file to the following item of [WorkUnit] > [WorkUnit name] > [Environment Settings] tab.

Path

Library Path

Notes about Using JNI with J2EE Applications

The same Native module cannot be loaded in different class loaders. If a class that uses JNI is contained in the application (EAR, ejb-jar, WAR, RAR) and the same Native module is loaded in a different class loader, java.lang.UnsatisfiedLinkError is thrown.

If java.lang.UnsatisfiedLinkError is thrown, take the following action.

- Set the class that uses JNI (the class that implements the 'native' method) in the WorkUnit class path without including it in the application, or save it in the IJServer 'Shared' directory.
- If the class that uses JNI is an EJB application class, it is possible to avoid the error by selecting 'Do not Use' for the class loader separation.

Additionally, if a class that uses JNI is contained in the application (EAR, ejb-jar, WAR, RAR), that application cannot use HotDeploy/class auto reload.

If an attempt to use HotDeploy/class auto reload is made, it may cause java.lang.UnsatisfiedLinkError to be thrown, and HotDeploy/class auto reload to fail. If this happens, restart IJServer.

To use HotDeploy/class auto reload in an application that uses JNI, take the following action.

- Set the class that uses JNI (the class that implements the 'native' method) in the WorkUnit class path without including it in the application, or save it in the IJServer 'Shared' directory.

IJServer must be restarted if the class that uses JNI is switched.

Transaction Control

This section explains the IJServer's J2EE application transaction control.

Transaction Control Method

To perform transaction control in the Web application, acquire UserTransaction from JNDI.

To perform transaction control in the EJB application, the user can specify a Bean (Bean Managed Transaction) as the transaction type with UserTransaction or specify a Container (Container Managed Transaction) as the transaction type so that the container controls the transaction.

Refer to 'Transaction Function using the UserTransaction Interface' in Chapter 3 for details of using UserTransaction.

As the transaction, both default and distributed transactions can be selected.

Default Transaction

A J2EE application on the same JavaVM can access the database with transaction linkage using the default transaction.

After the transaction is started, the transaction can be shared if the same data source is accessed via the same user/password, which enables the transaction linkage.

When a J2EE application starts the transaction and accesses another on the same JavaVM, it can be operated in the same transaction.

When an EJB application is transaction-linked, specify 'Container' as the transaction type and then specify a transaction linkage enabled transaction type (such as 'Required').

Distributed Transaction

When the following transaction linkages are made, use the distributed transaction.

- Transaction linkage by access to a different resource
- Transaction linkage between different IJServers
- Transaction start/end control from J2EE application client

Notes

- Because the distributed transaction can be used only with an EJB application, specify one of the following two as the IJServer operation type.
  - Process where only EJB applications are operated
  - Operating both Web and EJB applications (on separate JavaVMs)

  Specify the distributed transaction when the IJServer is defined.

  Refer to Part IV, JTS/JTA Edition, for details on the distributed transaction.
- The distributed transaction cannot be used between the EJB applications linked with different server machines.

- When transaction attributers, NotSupported, Supports, and Never, are specified in one of the Entity Bean methods by using the distributed transaction function, an error occurs when the objective EJB application is started, and the startup fails.
- IJServers cannot concurrently be started.
- Up to 32 resource managers can be used.

EJB Application Transaction

In the EJB application transaction function, JDBC connections are cached in the transaction. Therefore, distributed applications can be controlled with one transaction.

Caching JDBC Connections in Transaction

In the ordinary JDBC application, the transaction is managed for each connection acquired in the getConnection method of the data source.

In the case of the EJB application, the connection acquired in the getConnection method is cached in the transaction by the container. When the getConnection method is re-executed for the same data source in the same transaction, the container returns the cached connection. Therefore, the processing of applications distributed can be controlled with one transaction.

Note

Connections acquired from different data sources are processed with different transactions. Even if each data source is for the same database, it is processed with each transaction. Therefore, if the application is constructed, the processing may stop.

Transaction Linkage Enabled Resources

The resources that can be controlled with the default and distributed transactions are as follows.

- **Default transaction**

  The following resource can be transaction-controlled.

  – JDBC data source

  For a JMS connection factory where a Message-driven Bean receives a message, specify 'Container' as the Message-driven Bean's transaction type and 'Required' as the transaction type to let the container control the transaction.

- **Distributed transaction**

  The following resources can be transaction-controlled.

  – JDBC data source

  – JMS connection factory

  – Connector connection factory

  When the distributed transaction is used, distributed-transaction linkage enabled resources must have been defined.

Chapter 2

Operating J2EE Applications

The Interstage Management Console provided in Interstage V6 or later allows the user to start/stop Interstage and IJServer (operation) or set up the environment.

This chapter explains the operations listed below that are required for J2EE application operation.

- Preparing J2EE Applications
- Deploying and Setting J2EE Applications
- Preparation for Servlet Service Operation
- Request Distribution Control by Web Server Connector
- Procedure for Using JTS
- Procedure for Using JMS
- Procedure for Using JavaMail
- Customizing and Checking the Operating Environment
- Application debug

Refer to the sections in the corresponding chapters for details of usage of the following functions:

- Refer to Chapter 3, JNDI for details of the resource definition/reference function using JNDI.
- Refer to Chapter 4, The J2EE Application Security Function for details of the security function.

Point

To prevent incompatibility problems attributable to different Java VM versions, it is recommended to use the same version of JDK/JRE for development, deployment, and operation.

Preparing J2EE Applications

This section explains how to prepare J2EE applications.

Developing J2EE Applications

J2EE applications need to be developed.

Apworks can be used to develop J2EE applications. Refer to the 'Apworks Apdesigner Programmer's Guide' or 'Component Designer User's Guide' (not provided by Plus Developer) for details.

Refer to the following for details on development:

- 'Referencing Objects' in Chapter 3, for referencing resources and EJB
- Chapter 6, Web Application Development, for Web application
- Chapter 9, EJB Application Development, for EJB application

  Refer to Chapter 10, How to Create Entity, and Chapter 11, How to Call EJB Applications, for details on development in each runtime environment.

Setting the Deployment Descriptor

To prepare J2EE application clients or Web application, the deployment descriptor must be set. Set information on the J2EE applications and the resource names to be referenced by the J2EE applications in the deployment descriptor. Refer to the following for details:

- 'J2EE Application Client deployment descriptor file Detailed Set Up' in Chapter 3, for J2EE application clients.
- 'Web Application Environment Definition File (Deployment Descriptor)' in Chapter 6, for Web applications.
- For EJB applications, use the Apdesigner EJB deployment descriptor editor of Apworks or the deployment descriptor file editor (*1) of Component Designer. Refer to the 'Apworks Apdesigner Programmer's Guide' or 'Component Designer User's Guide' for more information (*1).

  *1 Component Designer is not provided for Plus Developer.

Packaging Class Files

Class files created as programs are packaged.

- Store the J2EE application client in a JAR file for packaging.
- Store the Web application client in a WAR file for packaging.
- Store the EJB application client in a JAR file for packaging.

The above packages created for individual application types can further be packaged as an Enterprise Archive (EAR) file. Doing so enables all applications used for operation to be distributed as a single package.

Deploying and Setting J2EE Applications

This section explains the following topics:

- Deploying J2EE Applications
- HotDeploy Function of J2EE

Deploying J2EE Applications

Deploy J2EE applications in the runtime environment.

- Use the Interstage Management Console or ijsdeployment command for Web applications and EJB applications. If needed, create IJServer and deploy packaged applications.

  Refer to 'J2EE Operation Command' in the Reference Manual (Command Edition) for details of the ijsdeployment command.

- For J2EE application clients, copy the CLIENT-JAR file to the client runtime environment and use the jar command to decompress the deployment descriptor file in the CLIENT-JAR file into an arbitrary directory.

  If J2EE application clients are included in the EAR file, use the deployment function of the Interstage Management Console to expand them and then take out the CLIENT-JAR file from the decompressed files.

  Refer to 'IJServer file configuration' in Chapter 1 for details of the J2EE application client expansion destination.

Notes

- EJB application deployment runs Javac and therefore fails in a JRE environment. Install a JDK environment for this purpose.

HotDeploy Function of J2EE

If the J2EE HotDeploy function is used, modules can be deployed, redeployed, and undeployed without stopping IJServer. Web and EJB applications can also be added to IJServer in operation and these applications can also be updated or deleted.

Because the function enables requests to be issued to modules being deployed or not being undeployed, applications can be developed more efficiently and IJServer can be operated continuously.

The HotDeploy function is explained in the following order:

- Design Method
- Operation Method
- Status of Deployed Modules
- Modules that are Activated or Inactivated at Deployment, Redeployment, Undeployment, or Reactivating
- Shared Directory

Design Method

The 'HotDeploy function' and 'class auto-reload function' are offered to improve development efficiency and maintenance during operation. Although using just the HotDeploy function will improve efficiency, using the class auto-reload function as well will improve development efficiency even more. Refer to 'Class Auto-reload Function' for details of the class auto-reload function.

Operation Method

To use the HotDeploy function, in the Interstage Management Console, click [WorkUnit]. Click the [Create New] tab, click [Detailed Settings] and make settings in [Shared Definition]. Alternatively, after creating the WorkUnit, in the Interstage Management Console, click [WorkUnit] > 'WorkUnit Name'. Click the [Environment Settings] tab, and make the change in [Shared Definition].

The HotDeploy function can be used to deploy new modules and redeploy existing modules efficiently. For this reason, the IJServer start status may differ depending on whether or not the HotDeploy function is used.

The relationship between deployed modules and the IJServer start status is explained below.

- IJServer start status if there are no deployed modules

  IJServer will fail to start if there are no deployed modules, regardless of whether or not 'Use' or 'Do not Use' is selected for the HotDeploy function.

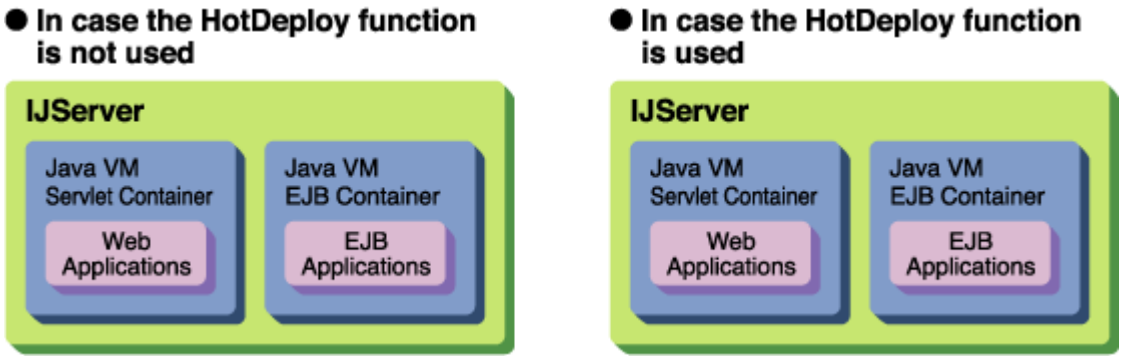
- IJServer start status if there are deployed modules

  If the IJServer type is 'Web and EJB applications are operated in different JavaVMs', the behavior is as shown below.

  All other IJServer types are as shown in the figure below, where the HotDeploy function is not used.

  – Deploying Web and EJB applications

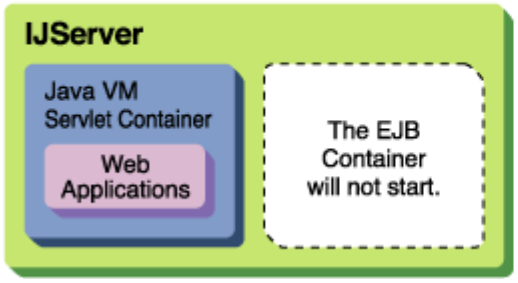
    All Java VMs (Servlet container and EJB container) start regardless of whether or not 'Use' or 'Do not Use' is selected for the HotDeploy function.



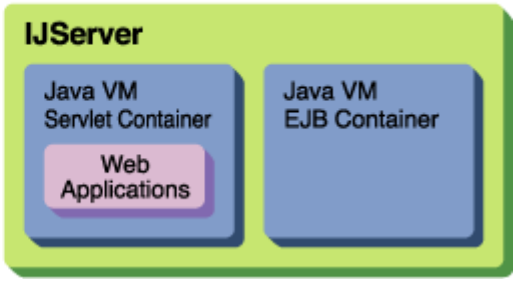
  – Deploying Web applications only

    Java VMs on which applications have been deployed start. If the HotDeploy function is used, all Java VMs (EJB container) start regardless of whether or not an EJB application has been deployed in an EJB container.

● In case the HotDeploy function is not used

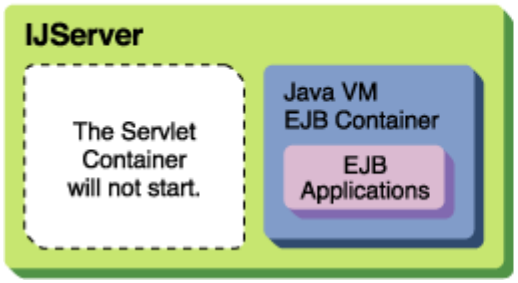


● In case the HotDeploy function is used

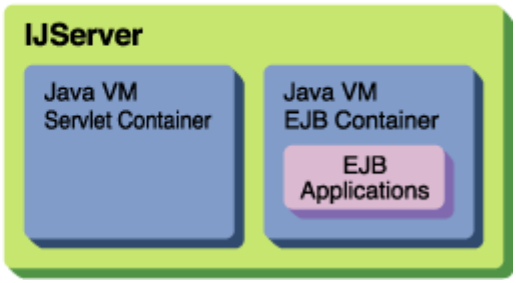


Java VMs on which applications have been deployed start. If the HotDeploy function is used, all Java VMs (Servlet container) start regardless of whether or not a Web application has been deployed in a Servlet container.

● In case the HotDeploy function is not used



● In case the HotDeploy function is used



- Start status when there are deployed modules but activation fails

  For EJB applications, the start status is as shown below. Web applications for which activation was successful start regardless of whether or not 'Use' or 'Do not Use' is selected for the HotDeploy function.

  – The HotDeploy function is used

    No EJB applications start.

  – The HotDeploy function is not used

    EJB applications for which activation was successful start.

  The table below shows the differences between using and not using the HotDeploy function.

[Separating between EARs/Separating all]

- If there are no deployment modules, IJServer does not start.
- If there are deployment modules, IJServer starts in the way shown in the table.

		The HotDeploy Function is not used		**The HotDeploy Function is used**	
Deployed Module Status/ Type		**No Modules can be Activated**	**All Modules can be Activated**	**No Modules can be Activated**	**All Modules can be Activated**
war		Starts		Starts	
ejb-jar		Does not start	Starts		
ear	war only	Does not start			
	ejb-jar only	Does not start	Starts		
	war and ejb-jar	Starts if all ejb-jar modules can be activated	Starts		

Note

If the HotDeploy function is not used, IJServer does not start if the EJB application activation fails.

Use the Interstage Management Console for deployment, redeployment, undeployment and reactivating.

1. Deployment (deployment of new modules)

   Modules are deployed to the operating environment and the modules deployed are activated.

   In the Interstage Management Console, click [WorkUnit] > 'WorkUnit Name' > [Deploy]. Click the [Browse] button, and select the target module for deployment.

2. Redeployment (redeployment of already deployed modules)

   Modules currently deployed are deactivated (*1), and the modules are redeployed and activated.

   In the Interstage Management Console, click [WorkUnit] > 'WorkUnit Name' > [Deploy]. Click the [Browse] button, and select the target module for redeployment.

3. Undeployment

   Modules currently deployed are deactivated and then undeployed.

   In the Interstage Management Console, click [WorkUnit] > 'WorkUnit Name'. Click the [Application Status/Undeploy] tab, and then click [Checkbox] to undeploy the selected deployed modules.

4. Reactivating (*2)

   Select [WorkUnit] > [IJServer name] > [Application Status/Undeploy] from the Interstage Management Console, and select a deployed module and click the Reactivate button. The module is deactivated to execute the re-reading of the definition file and destroy class files that have already been read.

   If reactivate is executed, the following settings are reflected.

   [For a Web application]

   – Module environment settings window settings

   – Module name conversion settings window settings

   [For a Web application]

   – Module environment settings window settings

   – Module name conversion settings window settings

   – Application environment definition window settings for applications

*1) The following operations are executed to deactivate a module already deployed:

   1) Stopping acceptance of additional requests

   2) Waiting until processing of the requests already accepted before start of deactivation processing is finished.

   3) Deactivation of the module

   For deactivation of a deployed module, acceptance of additional requests is stopped and deactivation processing is made to wait until processing for the current requests is finished. If request processing is not finished within one minute, an error occurs with the deployed module left in 'deactivation in progress'. In this case, wait until request processing is finished (the status is set to 'inactive'), and then reexecute deployment, or reboot IJServer.

   A module that has been put in 'inactive' state can be activated by reactivating it. A module that has been put in error state can be recovered for operation by removing the error cause and reactivating it.

   Refer to 'Status of deployed modules' for the status of deployed modules. Refer to' 4. Reactivating' for the reactivation procedure.

*2) Use the reactivate function to:

   – Activate the deployed module after removing the error cause for a module in the error state.

   – Change the tuning parameters or operation modes of a specific module without stopping IJServer.

   – Clear the cache of a specific module without stopping IJServer.

     For instance, clear the cache of the Entity Bean instance in instance management mode 'ReadOnly.' (When a module is deactivated, the application is initialized and therefore the information retained by the application and container is cleared.)

Status of Deployed Modules

The status of each module can be checked with the Interstage Management Console. The Interstage Management Console displays status information as shown below.

Status	Explanation	Action
Active	The deployed module can accept a request.	-
Inactive	The deployed module can accept no request.	Refer to the container log, establish the cause of the deactivated status and reactivate.
Active (part)	The deployed module can accept a request but is inactive for some IJServer processes.	Refer to the container log, establish the cause of the deactivated status and reactivate.
Active (inconsistent)	The deployed module can accept a request but the activated module is inconsistent among processes. (Because some IJServer processes are rebooted during deployment, certain processes have loaded modules before execution of redeployment.) This status is also generated when modules in 'active (part)' and 'active (inconsistent)' coexist.	-
Activating	The deployed module is going to start to accept requests.	-
Deactivating	The deployed module is going to stop accepting requests.	If there is no change in the status, it may be that processing was interrupted because of insufficient memory. Stop IJServer and then restart it.
Abnormal	Because some IJServer processes were rebooted during deployment, undeployment or reactivating, the following abnormal conditions are present: - Processes being activated and processes being deactivated coexist.	Stop IJServer and then restart it.

Modules that are Activated or Inactivated at Deployment, Redeployment, Undeployment, or Reactivating

When deploy, redeploy, undeploy, or reactivate is executed, deployed modules are deactivated or activated. In this case however, all the modules that reference the class of the deployment module are also deactivated or activated.

The reference relationships between deployment modules depend on the class loader separation method (separate between EARs, separate all, don't separate).

The table below summarizes deployment modules that are activated and deactivated depending on the class loader separation method. Refer to 'Separation of class loaders' in Chapter 1 for details of class loader separation.

Class Loader Setting	**Deployed Module**			
	EAR	**WAR**	**ejb-jar**	**RAR**
Separate between EARs	Target EAR only	Target WAR only	Every deployed ejb-jar, WAR or RAR	Every deployed ejb-jar, WAR or RAR
Separate all	Target EAR only	Target WAR only	Target ejb-jar only	Target RAR only
Don't separate	- (*)	- (*)	- (*)	- (*)

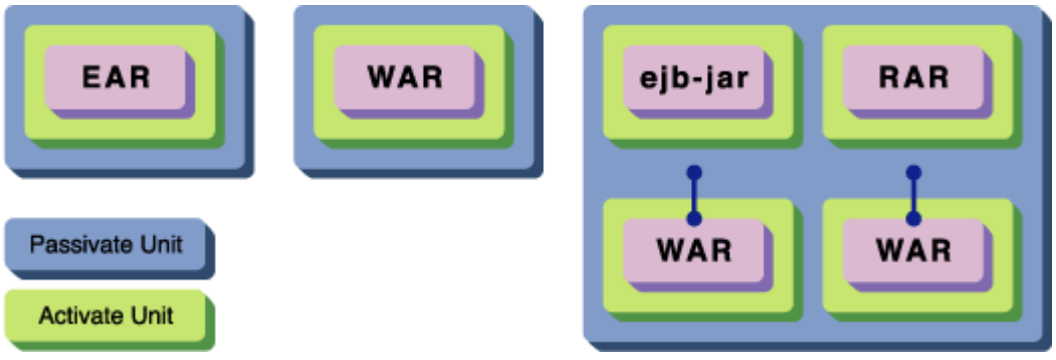
*: The HotDeploy function cannot be used.

Deactivation and activation according to the class loader separating system is explained below.

Deactivate/activate each module according to the range shown in the figure below. If the module activation fails, activation continues for all modules except the one that failed.

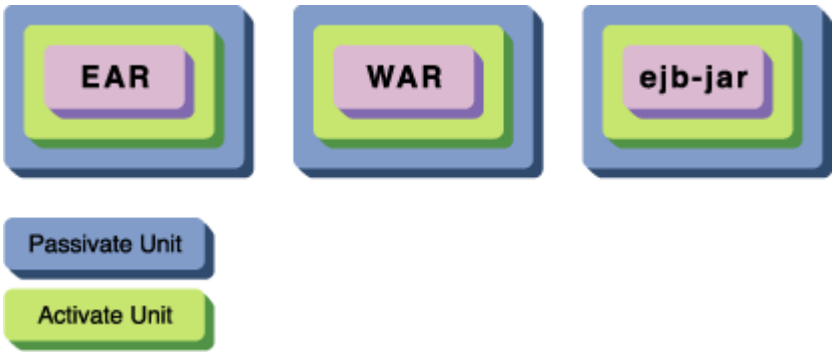
- Separating between EARs

  If the system used is separating between EARs, and an ejb-jar or RAR is deployed so that an ejb-jar or RAR class can be referenced from another ejb-jar, RAR, or WAR, all deployed ejb-jars, RARs, and WARs are deactivated/activated. If EAR or WAR is deployed, deployed modules are deactivated/activated individually.



- Separating all

  If the system used is separating all, modules are deactivated/activated individually.



Shared Directory

The Shared directory types are shown below. The reactivating behavior for each is different.

- Shared directory under the IJServer directory

  This is a directory in which classes are set for shared use in IJServer. Since class or jar files in this directory are not reactivated, if they are overwritten this operation does not take effect until IJServer is restarted.

- Shared directory contained in EAR

  This is a directory in which classes are set for shared use between applications in EAR. Class or jar files in this directory are targets of reactivation.

Notes about the HotDeploy function

- Changes made in the Interstage Management Console system, resources or definitions specified in IJServer are not effective.
- The Servlet session and STATEFUL Session Bean instance are destroyed. For this reason, re-create the files.
- When reactivating or redeploying an application containing a class that uses JNI, loading of a native module may fail, causing java.lang.UnsatisfiedLinkError to be thrown. In this case, it is necessary to restart IJServer. For details of executing HotDeploy for applications that use JNI, refer to 'Notes about using JNI with J2EE applications' in 'Notes about using class loaders'.

Class Auto-reload Function

The class auto-reload function is used to switch deployed application classes without stopping IJServer.

However, class auto-reload does not work when new if jar files are added. For this reason, jar files that are added are not reactivated until the module is reactivated or IJServer is restarted.

To make the settings for using the class auto-reload function, in the Interstage Management Console click [WorkUnit] > [Deploy]. Select 'Use' for the class auto-reload function.

If the class auto-reload function is used, a corrected application file is loaded automatically just by overwriting it. Development efficiency is improved because there is no need to redeploy the application or stop/start IJServer.

It is recommended that the class auto-reload function is executed for the development of applications.

When the auto-reload setting is changed while IJServer is active, the class that is changed is loaded according to one of the following operations:

- Re-start of IJServer
- Reactivation of module

Note

Requests to modules other than the one being auto-reloaded can be processed.

Design

The class auto-reload function can improve the efficiency of application development that requires class files of a deployed module be frequently modified and verified for operation. However, the function deteriorates processing performance because the container keeps monitoring class files for modification. For this reason, use the class auto-reload function only for application development.

In addition, the auto-reload function cannot replace the following classes. To replace these classes, use the HotDeploy function. Refer to 'HotDeploy Function of J2EE' for details of the HotDeploy function.

[Classes that the class auto-reload function cannot replace]

EJB interfaces (Remote, Home, Local, and LocalHome)

Operation

Because the class auto-reload function periodically monitors class files, the monitoring intervals must be defined from the Interstage Management Console. From the Interstage Management Console, select [WorkUnit] > [IJServer name] > [Environment Settings] and make settings for the deployed modules

To actually replace a class file, copy it directly to the deployment directory. Classes contained in the directories shown below are targets of the class auto-reload function. Refer to 'IJServer file configuration' in Chapter 1 for details of the copy destination (deployment directory).

If WAR files are deployed

The extension for files under [J2EEshareddirectory]\ijserver\[IJServer name]\apps\[Webmodule name]\WEB-INF\lib is '.jar'

[J2EE common directory]\ijserver\[IJServer name]\apps\[EJBmodule name]\lib is '.class'

If ejb-jar files are deployed

[J2EE common directory]\ijserver\[IJServer name]\apps\[EJBmodule name]\lib is '.class'

If EAR files are deployed

[J2EE common directory]\ijserver\[IJServer name]\apps\[EAR module name]\[WAR file name]\WEB-INF\lib is '.jar'

[J2EE common directory]\ijserver\[IJServer name]\apps\[EAR module name]\[ejb-jar file name]lib is '.class'

[J2EE common directory]\ijserver\[IJServer name]\apps\[EAR module name]\[RAR file name]lib is '.jar'

[J2EE common directory]\ijserver\[IJServer name]\apps\[EAR module name]\Shared\lib\lib is '.jar'

[J2EE common directory]\ijserver\[IJServer name]\apps\[EAR module name] \Shared\classes\lib is '.class'

Solaris OE Linux

If WAR files are deployed

/opt/FJSVj2ee/var/deployment/ijserver/[IJServer name]/apps/[Web module name]/WEB-INF/lib is '.jar'

/opt/FJSVj2ee/var/deployment/ijserver/[IJServer name]/apps/[Web module name]/WEB-INF/classes lib is '.class'

If ejb-jar files are deployed

/opt/FJSVj2ee/var/deployment/ijserver/[IJServer name]/apps/[EJB module name]/ lib is'.class'

If EAR files are deployed

opt/FJSVj2ee/var/deployment/ijserver/[IJServer name]/apps/[EAR module name]/[WAR file name]/WEB-INF/lib is'.jar'

/opt/FJSVj2ee/var/deployment/ijserver/[IJServer name]/apps/[EAR module name]/[WAR file name]/WEB-INF/lib is '.class'

/opt/FJSVj2ee/var/deployment/ijserver/[IJServer name]/apps/[EAR module name]/[ejb-jar file name]/lib is '.class'

/opt/FJSVj2ee/var/deployment/ijserver/[IJServer name]/apps/[EAR module name]/[RAR file name]/lib is '.jar'

/opt/FJSVj2ee/var/deployment/ijserver/[IJServer name]/apps/[EAR module name]/Shared/lib is '.jar'

/opt/FJSVj2ee/var/deployment/ijserver/[IJServer name]/apps/[EAR module name]/Shared/classes/lib is '.class'

When a class file is replaced, the classes of the modules that can reference the class are all auto-reloaded. The modules that can reference class files depend on the class loader settings. The classes of the modules that are auto-reloaded are shown below:

[Classes that are class auto-reload]

Class Loader Setting	**Classes to be Replaced**						
	Class of WAR Deployed Individually	**Class of WAR included in EAR**	**Class of WAR included in EAR**	**Class of ejb-jar included in EAR**	**Class of Shared Directory included in EAR**	**Class of RAR Deployed Individually**	**Class of RAR included in EAR**
Separate between EARs	Only class of replaced WAR	Only class of replaced WAR	All classes of ejb-jar, RAR, and WAR deployed individually	All classes of modules included in EAR	All classes of modules included in EAR	- (1)	All classes of modules included in EAR
Separate all	Only class of replaced WAR	Only class of replaced WAR	Only class of replaced ejb-jar	All classes of modules included in EAR	All classes of modules included in EAR	- (1)	All classes of modules included in EAR

Class Loader Setting	Classes to be Replaced						
	Class of WAR Deployed Individually	Class of WAR included in EAR	Class of WAR included in EAR	Class of ejb-jar included in EAR	Class of Shared Directory included in EAR	Class of RAR Deployed Individually	Class of RAR included in EAR
Don't separate	Only class of replaced WAR	- (1)	- (1)	- (1)	- (1)	- (1)	- (1)

1) Because the class loader is not separated, all deployed modules can be referenced to each other, except for those between WARs. For this reason, the class auto-reload function does not work.

2) Lone RARs that are deployed are not targets of auto-reload. For this reason, class auto-reload does not work.

Shared Directory

The Shared directory types are shown below. The auto-reload behavior for each is different.

- Shared directory under the IJServer directory

  This is a directory in which classes are set for shared use in IJServer. Since class or jar files in this directory are not auto-reloaded, if they are overwritten this operation does not take effect until IJServer is restarted.

- Shared directory contained in EAR

  This is a directory in which classes are set for shared use between applications in EAR. Class or jar files in this directory are auto-reload targets.

Notes about the auto-reload function

- In order to change the resources in the deployment directory of the application directly, users must have the appropriate level of authority. An administrator may need to change a general user's authority.
- The EJB interfaces (Remote, Home, Local, and LocalHome) and the resource adapter interface cannot be changed. If an interface is changed, one of the following errors may occur. If so, use the HotDeploy function for redeployment.
  - NoClassDefFoundException
  - NoClassDefFoundError
  - NoSuchMethodError
  - In CMP2.0 Entity Bean, the [CMP2.x-XXXX] message may be output.
- A module is not auto-reloaded when it is inactive.
- The class auto-reload function deteriorates processing performance because the container keeps monitoring class files for modification. For this reason, use the class auto-reload function only for application development.
- When the date and time of the update of the JSP file is newer than the class file generated by the last compile, JSP is auto-reloaded.

- The class auto-reload function only switches classes. For this reason, the result of changing a definition such as deployment descriptor in the Interstage Management Console system, resources or definitions specified in IJServer is invalid.
- The interstage.xml stored in the application deployment directory is used for running IJServer. For this reason, it must not be deleted. Additionally, if interstage.xml is edited using a text editor, only edit the <web> and <ejb> tags. If interstage.xml is deleted, or any tags other than the <web> and <ejb> tags are edited, it will affect the ability of IJServer to run normally. In this case, the corresponding IJServer must be deleted.

  For details of the application deployment directory, refer to 'IJServer file configuration'.

  For details of how to edit interstage.xml, refer to 'interstage.xml file'.
- If class auto-reload is used, whether or not application deactivation/activation works depends on the class file and jar file switch, as shown in the table below.

	A class File is added	**A jar File is added**	**A class File that has already been loaded is overwritten**	**A class file that has not been loaded is overwritten**	**A jar file is overwritten**
Application Deactivation/ Activation	Does not work	Does not work	Works	Does not work	Works

  When deactivation/activation is running for an application, the Servlet session and the STATEFUL Session Bean instance are destroyed. For this reason, they must be re-created.
- When class auto-reload is performed for an application containing a class that uses JNI, loading of a native module may fail, causing java.lang.UnsatisfiedLinkError to be thrown. In this case, it is necessary to restart IJServer. For details of executing class auto-reload for applications that use JNI, refer to 'Notes about using JNI with J2EE applications' in 'Notes about using class loaders'.

- When a file is copied, set the original access permissions also to the copied file. If invalid access permissions are set, application execution, deployment, or undeployment may fail.

Customizing Web and EJB Applications

Use the Interstage Management Console to customize Web and EJB applications.

From the Interstage Management Console, select [System] > [WorkUnit] > [IJServer name], and then click the application to be customized.

Setting Clients

To allow J2EE application clients and applets to reference EJB, the EJB client environment must be set up. Refer to 'Environment Setup for Referencing EJB' in Chapter 3 for details.

HTTP Tunneling

When using J2EE HTTP tunneling, refer to 'HTTP Tunneling of J2EE' in the Security System Guide.

Preparation for Servlet Service Operation

This section explains how to prepare for Servlet service operation.

Setting up Web Server Environment

The following Web server supports a connection with the Servlet service:

- Interstage HTTP Server

Windows

- Microsoft® Internet Information Services

Solaris OE

- Sun Java System Web Server

Interstage HTTP Server Environment Settings

The Web server connector implements communication from the Web server to the Servlet service.

The Web server connector operates as a DSO (dynamic shared objects) module using Apache API on the Web server, and therefore interlocks with the start and stop of the Web server.

If IJServer and Web server are operated on separate machines, the Web server connector environment must be set up. Refer to 'Procedure for Operation by Separating IJServer and Web Server' for the procedure for operation by separating IJServer and Web server.

Point

The definition information of the Web server connector shown below is set at the end of the environment definition file (httpd.conf) of the Interstage HTTP Server. If the environment definition file (httpd.conf) of the Interstage HTTP Server is to be edited without using the Interstage Management Console, do not delete the definition information or do not move it from the end of the file.

Windows

```
LoadModule jk2_module "C:/Interstage/F3FMjs4/gateway/mod_jk2.dll"
```

Solaris OE Linux

```
LoadModule jk2_module "/opt/FJSVjs4/gateway/mod_jk2.so"
```

Microsoft® Internet Information Services Environment Settings

Windows

The Microsoft® Internet Information Services Web server connector runs on the Web server using an embedded ISAPI API as an ISAPI filter and an ISAPI extension. For this reason, start and stop of the Web server are linked.

To separate and operate IJServer and the Web server on different machines, appropriate environment settings must be made in the Web server connector. For details, refer to 'Procedure for Operation by Separating IJServer and Web Server'.

Point

In the Web server connector of Microsoft® Internet Information Services, it is possible to reference the settings of the Web server connector of Interstage HTTP Server to run the connector in the same way. For this reason, even if the Web server connector of Microsoft® Internet Information Services is used, in the same way as for using Interstage HTTP Server, it is necessary to make the Interstage HTTP Server and Web server connector settings using the Interstage Management Console.

Note

Interstage HTTP Server and Microsoft ® Internet Information Services cannot use a Web server connector simultaneously, although it is possible for them to exist together by setting up a different port number in each Web server.

Link Interstage and Microsoft® Internet Information Services according to the following procedure:

1. Installing Microsoft® Internet Information Services and Interstage
2. Preventing Interstage HTTP Server Automatic Startup
3. Microsoft® Internet Information Services Environment Settings
4. Interstage Environment Settings

Installing Microsoft® Internet Information Services and Interstage

Install Microsoft® Internet Information Services and Interstage in the server machine.

Point

Interstage HTTP Server must be installed before Microsoft® Internet Information Services can be used.

If Interstage HTTP Server is not installed, it cannot be linked with Microsoft® Internet Information Services.

Preventing Interstage HTTP Server Automatic Startup

This is executed to prevent automatic startup of Interstage HTTP Server.

1. Stop Interstage HTTP Server

   In the Interstage Management Console, click [Services] > [Web Server]. Click the [Status] tab, and then click the [Stop] button. This stops Interstage HTTP Server.

2. Prevent linkage of Interstage and Interstage HTTP Server

   In the Interstage Management Console, click [System]. Click the [Environment Settings] tab, and then click [Link with Interstage] > [Web Server]. Set this to [Do not Link].

3. Log in with Administrator authority. Click [Control Panel], and then [Services], or click [Control Panel] – [Administrative Tools] - [Services]. After startup, select 'FJapache', and then click the [General] tab. Change [Startup Type] to [Manual].

Microsoft® Internet Information Services Environment Settings

To run the Web server connector of Microsoft® Internet Information Services as an ISAPI filter and an ISAPI extension, update the settings as shown below. The update uses the Internet service manager of Microsoft® Internet Information Services.

- Register the Web server connector so that it runs as an ISAPI filter and an ISAPI extension.

To set an access restriction in a Web application, define the security environment.

For details of the registering procedure, refer to the manual for Microsoft® Internet Information Services.

Note

The Microsoft® Internet Information Services environment settings are not the target of backup/restore or export/import.

Stopping Microsoft® Internet Information Services

If Microsoft® Internet Information Services has started up, stop it. To stop Microsoft® Internet Information Services, log in as a user with Administrator authority. Click [Control Panel], and then [Services], or click [Control Panel] – [Administrative Tools] - [Services]. After startup, select 'World Wide Web Publishing Service', and then select [Operate]. Next, select [Stop] from the list.

Registering in an ISAPI Filter

Set the Web server connector in 'ISAPI filter' of the Web site.

Set the following Web server connector file name in the filter file name.

```
C:\Interstage\F3FMjs4\gateway\isapi\isapi_redirector2.dll
```

Registering in an ISAPI Extension

Create a 'virtual directory' in the Web site.

The values that are set are shown below.

Local path	C:\Interstage\F3FMjs4\gateway\isapi
Name (alias)	F3FMjs4

Allow permission to execute access authority to the virtual directory.

Access Restriction Settings

To set an access restriction in a Web application, set the access restriction for the Web server connector.

Make the settings in 'directory security' of 'virtual directory'.

Starting up Microsoft® Internet Information Services

To start up Microsoft® Internet Information Services, log in with Administrator authority. Click [Control Panel], and then [Services], or click [Control Panel] - [Administrative Tools] - [Services]. After startup, select 'World Wide Web Publishing Service', and then select [Operate]. Next, select [Start] from the list.

Note

To use Microsoft® Internet Information Services 6.0, run it on an IIS5.0 process separation mode.

Interstage Environment Settings

The operations for making Interstage environment settings (such as creating a WorkUnit, or deploying a Web application) using the Interstage Management Console and using Interstage HTTP Server are the same.

However, the following differences exist between the operations in Interstage HTTP Server and Microsoft® Internet Information Services.

- If Microsoft® Internet Information Services is used, the Web server connector cannot be used for multiple Web sites on the same machine. Also, the Web server virtual host cannot be used as the Interstage HTTP Server in the Interstage Management Console.
- The Web server settings are made using the Internet service manager that is offered by Microsoft® Internet Information Services.

Note

To use SSL communication between the Web server connector and the Servlet container, the user who executes Microsoft® Internet Information Service must have been granted permission to access the Interstage certificate environment, and must own Administrators authority. To use the SSL function as a user with general authority, select the Interstage certificate environment folder in Explorer, and add access authority for the user or group using the [Properties] menu, [Security] tab window. Set [Full Control] for the user or group that is added.

For details about access authority Interstage certificate environment, refer to the Security System Guide. The relevant section is 'Interstage certificate environment access authority settings' in 'Setting up and using the Interstage certificate environment'.

Sun Java System Web Server Environment Settings

The Web server connector of Sun Java System WebServer is run on the Web server using NSAPI API as a plug-in. For this reason, start and stop of the Web server are linked.

To separate and operate IJServer and the Web server on different machines, appropriate environment settings must be made in the Web server connector. For details, refer to 'Procedure for Operation by Separating IJServer and Web Server'.

Point

In the Web server connector of Sun Java System WebServer, it is possible to reference the settings of the Web server connector of Interstage HTTP Server to run the connector in the same way. For this reason, even if the Web server connector of Sun Java System WebServer is used, in the same way as for using Interstage HTTP Server, it is necessary to make the Interstage HTTP Server and Web server connector settings using the Interstage Management Console.

Note

Interstage HTTP Server and Sun Java System WebServer cannot use a Web server connector simultaneously, although it is possible for them to exist together by setting up a different port number in each Web server.

Link Interstage and Sun Java System WebServer according to the following procedure:

1. Installing Sun Java System WebServer and Interstage
2. Preventing Interstage HTTP Server Automatic Startup
3. Sun Java System Web Server Environment Settings
4. Interstage Environment Settings

Installing Sun Java System WebServer and Interstage

Install Sun Java System WebServer and Interstage in the server machine.

Point

Interstage HTTP Server must be installed before Sun Java System WebServer can be used.

If Interstage HTTP Server is not installed, it cannot be linked with Sun Java System WebServer.

Note

When installing Sun Java System Web Server, specify 'nobody' for the user and group that are used to execute the default instance. Do not specify 'webservd'.

If you are using a user and group other than 'nobody', make sure that they are the same as the 'User' and 'Group' defined in the Interstage HTTP Server environment definition file. For details of the Interstage HTTP Server environment definition file, refer to 'Environment Definition File' in the Interstage Application Server Web Server Operator’s Guide (Interstage HTTP Server Edition).

Preventing Interstage HTTP Server Automatic Startup

This is executed to prevent automatic startup of Interstage HTTP Server.

1. Stop Interstage HTTP Server

   In the Interstage Management Console, click [Services] > [Web Server]. Click the [Status] tab, and then click the [Stop] button. This stops Interstage HTTP Server.

2. Prevent linkage of Interstage and Interstage HTTP Server

   In the Interstage Management Console, click [System]. Click the [Environment Settings] tab, and then click [Link with Interstage] > [Web Server]. Set this to [Do not Link].

3. Prevent execution of the start shell and stop shell

   Evacuate the symbolic link file for the Interstage HTTP Server start and stop shells. The start shell is called when the server machine starts up, but the stop shell call is prevented when the server machine stops.

   The target files are as shown below.

   - /etc/rcS.d/K17FJapache
   - /etc/rc0.d/K17FJapache
   - /etc/rc1.d/K17FJapache

- /etc/rc2.d/K17FJapache
- /etc/rc3.d/S51FJapache

Example

```
mv /etc/rcS.d/K17FJapache /etc/rcS.d/_K17FJapache
mv /etc/rc0.d/K17FJapache /etc/rc0.d/_K17FJapache
mv /etc/rc1.d/K17FJapache /etc/rc1.d/_K17FJapache
mv /etc/rc2.d/K17FJapache /etc/rc2.d/_K17FJapache
mv /etc/rc3.d/S51FJapache /etc/rc3.d/_S51FJapache
```

Sun Java System Web Server Environment Settings

To run the Web server connector as a Sun Java System Web Server plug-in module, edit the Sun Java System Web Server magnus.conf, obj.conf and mime.types as shown below.

Use a text editor to edit them.

Storage Directory

magnus.conf, obj.conf and mime.types are stored in the following directory:

server-root/server-id/config/

server-root indicates the Sun Java System Web Server installation directory, and server-id indicates the ServerID of each server. For example, if the Sun Java System Web Server installation directory is '/opt/SUNWwbsvr', and the ServerID is 'https-taro', magnus.conf and obj.conf are stored in the following directory:

/opt/SUNWwbsvr/https-taro/config

magnus.conf settings

In the StackSize directive, set a value of at least 262144 (256K).

```
Stack Size 262144
```

In the ChildRestartCallback directive, set 'on', 'yes', or 'true'.

```
ChildRestartCallback on
```

Set the maximum number of processes that can be executed at the same time (MaxProcs), and the maximum number of threads that can be processed at the same time in each process (RqThrottle) in Sun Java System Web Server.

The number of requests that Sun Java System Web Server can process at the same time is calculated according to a formula that multiplies MaxProcs and RqThrottle. For this reason, MaxProcs and RqThrottle must be set so that they satisfy the following conditions:

```
MaxProcs * RqThrottle <= number of Servlet containers that can be processed
at the same time
```

For example, if the number of Servlet containers that can be processed at the same time is set as 64, set the value of MaxProcs × RqThrottle as 64 or less.

Also, it is recommended that the value for MaxProcs is '1' if the machine that is used has a single processor (CPU). For details, refer to the Sun Java System Web Server manual.

```
MaxProcs 1
RqThrottle 64
```

Add the 2 directives to the last line.

```
Init fn="load-modules" funcs="ijs_nsapi_init,ijs_nsapi_handler"
shlib="/opt/FJSVjs4/gateway/nsapi/ijs_nsapi_redirector.so"
Init fn="ijs_nsapi_init" conf="/opt/FJSVihs/conf/workers2.properties"
```

obj.conf settings

Add 1 directive to the line after the '<Object name=default>' tab.

```
<Object name=default>
Service fn="ijs_nsapi_handler"
...
</Object>
```

Disabling the Sun Java System Web Server Default Servlet and JSP

In Sun Java System Web Server, the servlet and JSP that are offered by Sun Java System Web Server are set so that they can be run by default. These must be disabled before processing can occur using the Web server connector.

Sun Java System Web Server 6.0

In mime.types, either delete the line below, or make it a comment.

```
type=magnus-internal/jsp  exts=jsp
```

Sun Java System Web Server 6.1

In magnus.conf, either delete the line below, or make it a comment.

```
Init fn="load-modules"
shlib="/opt/SUNWwbsvr/bin/https/lib/libj2eeplugin.so"
shlib_flags="(global|now)"
```

In obj.conf, either delete the line below, or make it a comment.

```
NameTrans fn="ntrans-j2ee" name="j2ee"
Error fn="error-j2ee"
```

Interstage Environment Settings

The operations for making Interstage environment settings (such as creating a WorkUnit, or deploying a Web application) using the Interstage Management Console and using Interstage HTTP Server are the same.

However, the following differences exist between the operations in Interstage HTTP Server and Sun Java System Web Server.

- If Sun Java System Web Server is used, the Web server virtual host cannot be used.
- The Web server settings are made using the Administration Server that is offered by Sun Java System Web Server.

Note

To use SSL communication between the Web server connector and the Servlet container, the user who executes Sun Java System Web Server must have been granted permission to access the Interstage certificate environment.

For details about access authority Interstage certificate environment, refer to the Security System Guide. The relevant section is 'Interstage certificate environment access authority settings' in 'Setting up and using the Interstage certificate environment'.

Procedure for Operation by Separating IJServer and Web Server

IJServer and Web server can be operated on separate machines. This function enables system construction as follows:

- Security improvement

  If it is not desired to operate IJServer in the demilitarized zone (DMZ), operate the Web server on the DMZ machine and operate IJServer on a machine in the intranet inside the firewall.

- Load distribution

  When the CPU load of the machine is too much, construct a system in which IJServer and Web server are operated on separate machines to distribute the load.

  Using Traffic Director enables load distribution while monitoring the operating conditions and CPU load of the machine on which IJServer operates. When using Traffic Director, secure the necessary number of permanent connections (Web acceleration function) during setting the Traffic Director distribution ports. Match the number of permanent connections with the number of concurrent Servlet container processing tasks. Refer to the Interstage Traffic Director's Manual for details of the Traffic Director.

Connect IJServer and Web server under http or https.

The following explains the procedure for operating IJServer and Web server on separate machines, including the differences from that procedure for operating them on the same server machine.

Point

- Use the same version level of Interstage in IJServer and the Web server.
- To operate IJServer and Web server on separate machines, first make the following settings. Without these settings, the Web server connector cannot be operated.

  Using the Interstage Management Console on each of the IJServer and Web server machines, select [System] > [Environment Settings] tab and make settings so that Web server and IJServer machines are separated.
- To limit the Web server IP addresses for connecting to the Servlet container, and set more than one IP address in the Web server, click [WorkUnit] > 'WorkUnit Name' > [Environment Definition] tab > [Advanced Settings] of the Interstage Management Console. Next, click [Web Server Connector(Connector) Settings] > [Web Server IP Address], and specify all the IP addresses that are to be set in the Web server.

If the following settings are made in the IJServer machine and the Web server machine, IJServer and the Web server can be separated and operated.

1. Creating IJServer

   Create IJServer using the Interstage Management Console on the IJServer machine.

2. Setting connection destination IJServer

   Using the Interstage Management Console on the Web server machine, create Web server connector connection destination information based on the information specified at 'Creating IJServer.'

3. Deploying Web applications

   Using the Interstage Management Console on the IJServer machine, deploy Web applications to the IJServer created at 'Creating IJServer.'

4. Setting the connection destination Web application

   Using the Interstage Management Console on the Web server machine, set the name of the Web application deployed by 'Deploying Web applications' in the information of the connection destination IJServer created at 'Setting connection destination IJServer.'

If the following operations are executed in an IJServer machine, the above operations that are executed in the Web server machine must also be reflected in the IJServer machine.

- When changing the following items in [Web server connector (connector) setting] or [Servlet container setting] in the IJServer environment setup
  - [Web server connector (connector) settings]

    Using SSL between the connector and the Servlet container

    SSL definition between the connector and the Servlet container

- [Servlet Container Settings]

  Servlet container IP address

  Port number

  Timeout

  The number of simultaneous processings

  In the Interstage Management Console of the Web server machine, specify [Web Server] > [Web Server Connector]. Next, in the right frame click the name of the WorkUnit for which the change was executed in the IJServer machine. This reflects the value for the change that was executed in the IJServer machine.

- When deleting IJServer

  In the Interstage Management Console of the Web server machine, click [Web Server] > [Web Server Connector]. Click the [List] tab], and select the name of the WorkUnit to be deleted.

- When undeploying a Web application

  In the Interstage Management Console of the Web server machine, specify [Web Server] > [Web Server Connector]. Next, in the right frame click the name of the WorkUnit that was undeployed in the IJServer machine, and then delete the undeployed Web application.

Spreading Requests when using Sessions in Servlet and JSP

When using sessions in Servlet and JSP, requests from the client are spread to the Servlet container in which the session is created according to the spread control of the Web server connector.

The Web server connector identifies the Servlet container in which the session is created using the Servlet container identifier, and spreads the request to that Servlet container.

If the Servlet container is standalone, meaning that the Web server and WorkUnit are not run on the same machine, the following definitions can be made using the standalone Interstage Management Console:

- [Services]> [Web Server] > [Web Server Connector] > [Create New]
- [Services]> [Web Server] > [Web Server Connector] > 'WorkUnit Name' > [Environment Settings]

In multiservers and standalone servers in which the Web server and WorkUnit are run on the same machine, the Servlet container identifier is automatically assigned a number and is managed internally. For this reason, there is no need to concern yourself with Servlet container identifiers.

Example of Preparation for Operation

An example of the operation procedure for each of the following machine configurations is explained below.

- One IJServer Machine and one Web Server Machine (Two-server Machine Configuration)
- One IJServer Machine and two Web Server Machines (Three-server machine Configuration)
- Two IJServer Machines and one Web Server Machine (Three-server Machine Configuration)
- Two IJServer Machines, Two Web Server Machines, and one Load Balancing Machine (Five-server Machine Configuration)

If three or more IJServer machines are configured, create a WorkUnit on each IJServer machine and set connection destination information on the Web server machine. For this operation, refer to steps 1, 2, and 3 at 'Two IJServer machines and one Web server machine (three-server machine configuration)' [For newly creating IJServer].

If three or more Web server machines are configured, set connection destination information on the Web server machine. For this operation, refer to steps 2 and 3 at 'One IJServer machine and two Web server machine (three-server machine configuration)' [For newly creating IJServer].

One IJServer Machine and one Web Server Machine (Two-server Machine Configuration)

[Requirements]

Server machine A (IP Address: 172.16.30.1)	Web server must be running
Server machine B (IP Address: 172.16.30.2)	IJServer must be running

1. Setup of server machine B for IJServer.

   From the Interstage Management Console on server machine B, select [WorkUnit] > [Create New] tab and create a WorkUnit with the following settings.

Item Name	**Setting Value Example**
WorkUnit name	MyIJServer
IP address for the Web server	172.16.30.1
IP address for Servlet container	172.16.30.2
Port number	9000
Timeout	480
The number of concurrent processing tasks	64

2. Setup of server machine A for the Web server.

   From the Interstage Management Console on server machine A, select [Services] > [Web Server] > [Web Server Connector] > [Create New] tab and create connection destination information with the following settings.

Item Name	**Setting Value Example**
WorkUnit name	MyIJServer
IP address for Servlet container : Port number	172.16.30.2:9000
Timeout	480
The number of concurrent processing tasks	64

3. Web application is deployed in server machine B for IJServer.

   From the Interstage Management Console on server machine B, select [WorkUnit] > [WorkUnit name (example: MyIJServer)] > [Deploy] tab and deploy the Web application.

4. A Web application name is added to server machine A for the Web server.

   From the Interstage Management Console on server machine A, select [Services] > [Web Server] > [Web Server Connector]. In the right frame, click the name of the WorkUnit that was deployed in 3 above to add the name of the deployed Web application.

One IJServer Machine and two Web Server Machines (Three-server machine Configuration)

[Requirements]

Server machine A (IP Address: 172.16.30.1)	Web server must be running
Server machine B (IP Address: 172.16.30.2)	Web server must be running
Server machine C (IP Address: 172.16.30.3)	IJServer must be running

1. Setup of server machine C for IJServer.

   From the Interstage Management Console on server machine B, select [WorkUnit] > [Create New] tab and create a WorkUnit with the following settings.

Item Name	**Setting Value Example**
WorkUnit name	MyIJServer
IP address for the Web server	172.16.30.1 172.16.30.2
IP address for Servlet container	172.16.30.3
Port number	9000
Timeout	480
The number of concurrent processing tasks	64

2. Setup of server machine A for the Web server.

   From the Interstage Management Console on server machine A, select [Services] > [Web Server] > [Web Server Connector] > [Create New] tab and create connection destination information with the following settings.

   The number of concurrent processing tasks must be set so that the total number for the two Web server machines (server machines A and B) matches the number of concurrent processing tasks of the IJServer machine (server machine C).

Item Name	**Setting Value Example**
WorkUnit name	MyIJServer
IP address for Servlet container : Port number	172.16.30.2:9000
Timeout	480
The number of concurrent processing tasks	32

3. Setup of server machine B for the Web server.

   Do the same operation as in 2 using the Interstage Management Console on server machine B.

4. Web application is deployed in server machine C for IJServer.

   From the Interstage Management Console on server machine C, select [WorkUnit] > [WorkUnit name (example: MyIJServer)] > [Deploy] tab and deploy the Web application.

5. A Web application name is added to server machine A for the Web server.

   From the Interstage Management Console on server machine A, select [Services] > [Web Server] > [Web Server Connector]. In the right frame, click the name of the WorkUnit that was deployed in 4 above to add the name of the deployed Web application.

6. A Web application name is added to server machine B for the Web server.

   Do the same operation as in 5 using the Interstage Management Console on server machine B.

Two IJServer Machines and one Web Server Machine (Three-server Machine Configuration)

[Requirements]

Server machine A (IP Address: 172.16.30.1)	Web server must be running
Server machine B (IP Address: 172.16.30.2)	IJServer must be running
Server machine C (IP Address: 172.16.30.3)	IJServer must be running

1. Setup of server machine B for IJServer.

   From the Interstage Management Console on server machine B, select [WorkUnit] > [Create New] tab and create a WorkUnit with the following settings.

Item Name	**Setting Value Example**
WorkUnit name	MyIJServer
IP address for the Web server	172.16.30.1
IP address for Servlet container	172.16.30.2
Port number	9000
Timeout	480
The number of concurrent processing tasks	64

2. Setup of server machine C for IJServer.

   Do the same operation as in 2 using the Interstage Management Console on server machine C. However, specify 172.16.30.3 for IP address.

   [For deploying the same Web application to server machines B and C]

   Set the same WorkUnit name as that of server machine B. The environment setup of the WorkUnit must also be the same.

Item Name	Setting Value Example
WorkUnit name	MyIJServer
IP address for the Web server	172.16.30.1
IP address for Servlet container	172.16.30.3
Port number	9000
Timeout	480
The number of concurrent processing tasks	64

[For deploying different Web applications to server machines B and C]

Set a WorkUnit name different from that of server machine B.

Item Name	Setting Value Example
WorkUnit name	MyIJServer_2
IP address for the Web server	172.16.30.1
IP address	172.16.30.3
Port number	9000
Timeout	480
The number of concurrent processing tasks	64

3. Setup of server machine A for the Web server.

From the Interstage Management Console on server machine A, select [Services] > [Web Server] > [Web Server Connector] > [Create New] tab and create connection destination information with the following settings.

[For deploying the same Web application to server machines B and C]

Set the connection destination information of the WorkUnit name set on server machines B and C.

Item Name	Setting Value Example
WorkUnit name	MyIJServer
IP address for Servlet container : Port number	172.16.30.2:9000 172.16.30.3:9000
Timeout	480
The number of concurrent processing tasks	64

[For deploying different Web applications to server machines B and C]

Create connection destination information separately for server machines B and C. Set the same WorkUnit name as that of the connection destination server machine.

<Connection destination information separately for server machines B>

Item Name	Setting Value Example
WorkUnit name	MyIJServer
IP address for Servlet container : Port number	172.16.30.2:9000
Timeout	480
The number of concurrent processing tasks	64

<Connection destination information separately for server machines C>

Item Name	Setting Value Example
WorkUnit name	MyIJServer_2
IP address for Servlet container : Port number	172.16.30.3:9000
Timeout	480
The number of concurrent processing tasks	64

4. Web application is deployed in server machine B for IJServer.

   From the Interstage Management Console on server machine B, select [WorkUnit] > [WorkUnit name (example: MyIJServer)] > [Deploy] tab and deploy the Web application.

5. Web application is deployed in server machine C for IJServer.

   Do the same operation as in 4 using the Interstage Management Console on server machine C.

6. A Web application name is added to server machine A for the Web server.

   From the Interstage Management Console on server machine A, select [Services] > [Web Server] > [Web Server Connector]. In the right frame, click the name of the WorkUnit that was deployed in 4 above to add the name of the deployed Web application.

Two IJServer Machines, Two Web Server Machines, and one Load Balancing Machine (Five-server Machine Configuration)

[Requirements]

Server machine A (IP Address: 172.16.30.1)	Load balancing is running
Server machine B (IP Address: 172.16.30.2)	Web server is running
Server machine C (IP Address: 172.16.30.3)	Web server is running
Server machine D (IP Address: 172.16.30.4)	IJServer is running
Server machine E (IP Address: 172.16.30.5)	IJServer is running

1. Setup of server machine D for IJServer.

   From the Interstage Management Console on server machine D, select [WorkUnit] > [Create New] tab and create a WorkUnit with the following settings.

Item Name	Setting Value Example
WorkUnit name	MyIJServer
IP address for the Web server	172.16.30.2 172.16.30.3
IP address for Servlet container	172.16.30.4
Port number	9000
Timeout	480
The number of concurrent processing tasks	64

2. Setup of server machine E for IJServer.

   Do the same operation as in 1 using the Interstage Management Console on server machine E. However, specify 172.16.30.5 for IP address.

3. Setup of server machine B for the Web server.

   From the Interstage Management Console on server machine B, select [Web Server] > [Web Server Connector] > [Create New] tab and create connection destination information with the following settings.

   Make the settings so that the total number of simultaneous processing events for each Web server machine (server machines B and C) and each IJServer server machine (server machines D and E) is the same.

Item Name	Setting Value Example
WorkUnit name	MyIJServer
IP address for Servlet container : Port number	172.16.30.4:9000 172.16.30.5:9000
Timeout	480
The number of concurrent processing tasks	64

4. Setup of server machine C for the Web server.

   Do the same operation as in 3 using the Interstage Management Console on server machine C.

   Define IP address : Port number for the Servlet container using the same order as for the value set in server machine B. If the order is different, it might mean that the session is not inherited normally.

5. Setup of server machine A for load balancing device.

   Set the load balancing device for the Web server machine A.

   Refer to the load balancing device manual for details of the load balancing device settings method.

6. Web application is deployed in server machine D for IJServer.

   From the Interstage Management Console on server machine D, select [WorkUnit] > [WorkUnit name (example: MyIJServer)] > [Deploy] tab and deploy the Web application.

7. Web application is deployed in server machine E for IJServer.

   Do the same operation as in 6 using the Interstage Management Console on server machine E.

8. A Web application name is added to server machine B for the Web server.

   From the Interstage Management Console on server machine B, select [Services] > [Web Server] > [Web Server Connector]. In the right frame, click the name of the WorkUnit that was deployed in 6 above to add the name of the deployed Web application.

9. A Web application name is added to server machine C for the Web server.

   Do the same operation as in 8 using the Interstage Management Console on server machine C.

Coexistence with Version 5.1 or earlier Servlet Service

Version 5.1 or earlier Servlet services can run concurrently on a single server machine.

If Web applications having the same application identifier are deployed to both a Servlet services and version 5.1 or earlier Servlet services, the Web application on the Servlet service defined later in the environment definition file (httpd.conf) of Interstage HTTP Server runs.

Request Distribution Control by Web Server Connector

This section explains the control for the distributing of requests in the Web server connector. This control consists of control by commands and control by fault monitoring.

Distributing Procedure and Viewing the Status using the Commands

The Web server connector implements request distribution control over each IJServer WorkUnit.

The user can use the following commands to change or display the current request distribution control mode (whether to enable or disable the distribution function of the Web server connector) over IJServer WorkUnits.

For instance, when periodic maintenance of a Web application is performed or if a network failure or machine hardware error occurs, the relevant IJServer WorkUnit (or the relevant machine) can be excluded from the request distribution targets. It can be restored to the distribution targets after completion of maintenance or failure recovery. This function enables stable and continuous system operation.

- Distribution operation command (ijsdispatchcont)

  This command can be used to include an IJServer WorkUnit in the request distribution targets or exclude it from the targets by specifying the IP address or the IP address and port number.

  The setting made by the distribution operation command is retained even after the Web server is rebooted or it terminates abnormally. To change the distribution mode, enter another distribution operation command.

- Distribution mode display command (ijsprintdispatchcont)

  This command displays the distribution mode of each IJServer WorkUnit (IP-address: port-number WorkUnit-name).

Check in advance the IP address or 'IP address: port number' of the IJServer WorkUnit to be specified in the distribution operation command. Use the distribution mode display command for this purpose. The distribution operation command requires only the IP address or 'IP address: port number' displayed by the distribution mode display command.

Refer to the Reference Manual (Command Edition) for details of each command.

Note

Requests are distributed from the Web server connector to each IJServer WorkUnit according to the following method. However, if session management is used in a Web application, the following requests are distributed to the IJServer WorkUnit in which the session was created.

Requests are distributed to the IJServer WorkUnit with the lowest number of requests that are currently being processed.

Solaris OE Linux

Requests are distributed in round-robin method.

Examples of operational patterns are shown below. Various operational patterns are available depending on the server machine configuration and command parameter specification method.

- Pattern 1: Distribution control for each machine
- Pattern 2: Distribution Control for each IJServer WorkUnit(1)
- Pattern 3: Distribution Control for each IJServer WorkUnit(2)
- Pattern 4: Suppress for a connection to the IJServer WorkUnit

Pattern 1: Distribution control for each machine

For maintenance of a machine or if a machine hardware error occurs, distribution to a specific machine can be controlled as follows:

A machine (machine 1) with IP address 123.123.123.110 and a machine (machine 2) with IP address 123.123.123.111 each consists of two IJServer WorkUnits (IJServerA and IJServerB).

Web application ap101 is deployed to IJServerA and Web application ap102 is deployed to IJServerB.

Because machine 1 is to be stopped, IP address 123.123.123.110 needs to be excluded from the distribution targets.

```
ijsdispatchcont OFF 123.123.123.110
```

ap101 and ap102, which were distributed to machines 1 and 2 for load distribution, now run on machine 2 alone.

When machine 1 is restored, use the following command to include it again in the distribution targets.

```
ijsdispatchcont ON 123.123.123.110
```

Pattern 2: Distribution Control for each IJServer WorkUnit(1)

For maintenance of a Web application, distribution to a specific IJServer WorkUnit can be controlled as follows:

As with pattern 1, a machine (machine 1) with IP address 123.123.123.110 and a machine (machine 2) with IP address 123.123.123.111 each consists of two IJServer WorkUnits (IJServerA and IJServerB).

Web application ap101 is deployed to IJServerA and Web application ap102 is deployed to IJServerB.

Because IJServerA on machine 1 is to be stopped, it needs to be excluded from the distribution targets.

```
ijsdispatchcont OFF 123.123.123.110:9000
```

ap101, which was distributed to machines 1 and 2 for load distribution, now runs on machine 2 alone. ap102 remains distributed to machines 1 and 2.

When IJServerA on machine 1 is restored, use the following command to include it again in the distribution targets.

```
ijsdispatchcont ON 123.123.123.110:9000
```

Pattern 3: Distribution Control for each IJServer WorkUnit(2)

For maintenance of a Web application while the number of concurrent processes of an IJServer WorkUnit is two or more, distribution to the IJServer WorkUnit can be controlled as follows:

A machine (machine 1) with IP address 123.123.123.110 and a machine (machine 2) with IP address 123.123.123.111 each consists of two IJServer WorkUnits (IJServerA and IJServerB).

Suppose the number of concurrent processes of IJServerA is 2 and two Servlet containers (container α and container β) are active.

Web application ap101 is deployed to IJServerA and Web application ap102 is deployed to IJServerB.

Because IJServerA on machine 1 is to be stopped, it needs to be excluded from the distribution targets.

Execute as many distribution operation commands as there are Servlet containers.

```
ijsdispatchcont OFF 123.123.123.110:9000
ijsdispatchcont OFF 123.123.123.110:9001
```

ap101, which was distributed to containers α and β of machines 1 and 2, now runs on containers α and β of machine 2 alone. ap102 remains distributed to IJServerB on machine 1 and IJServerB on machine 2.

When IJServerA on machine 1 is restored, use the following command to include it again in the distribution targets.

```
ijsdispatchcont ON 123.123.123.110:9000
ijsdispatchcont ON 123.123.123.110:9001
```

Pattern 4: Suppress for a connection to the IJServer WorkUnit

For maintenance of a Web application while one machine is used for the IJServer WorkUnit, a connection to the IJServer WorkUnit can be suppressed as follows:

A machine (machine 1) with IP address 123.123.123.110 consists of two IJServer WorkUnits (IJServerA and IJServerB).

Web application ap101 is deployed to IJServerA and Web application ap102 is deployed to IJServerB.

Because IJServerA on machine 1 is to be stopped, it needs to be excluded from the distribution targets.

```
ijsdispatchcont OFF 123.123.123.110:9000
```

ap101, which had only IJServer as the distribution target, can no longer work for processing. ap102 can still work with IJServerB.

* If a Web application such as ap101 has no distribution target, HTTP status code 503 (Service Temporarily Unavailable) is returned to the Web browser when a request is issued.

When IJServerA on machine 1 is restored, use the following command to include it again in the distribution targets.

```
ijsdispatchcont ON 123.123.123.110:9000
```

Monitoring Web Server Connector Faults

Note

This function can be used with the following products.

- Interstage Application Server Enterprise Edition
- Interstage Application Server Standard Edition
- Interstage Application Server Plus

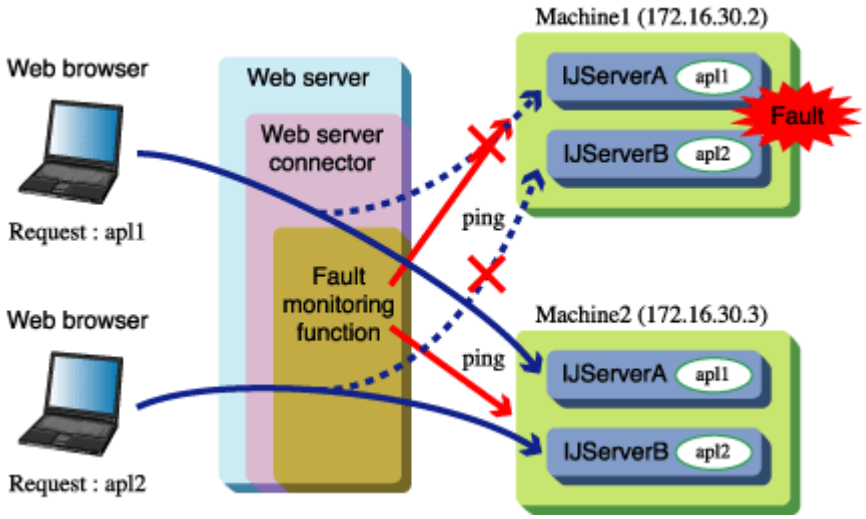
When a balancing operation is executed for IJServer and the Web server that have been separated on different server machines, you can monitor the operation status of the IJServer machine and the Servlet container to stop an IJServer machine or a faulty Servlet container from being distributed automatically, or to make it possible for an IJServer machine or a Servlet container that has recovered from a fault to be distributed automatically.

When the Web server connector fault monitoring function is used to stop an IJServer machine or a Servlet container from being distributed, the Web server connector stops the distributing of requests.

The method of fault monitoring function are as follows:

- ping monitoring

  This issues a ping (ICMP ECHO) to the IP address of the IJServer machine, and monitors the operation status while checking for the presence of responses (ICMP ECHO REPLY).

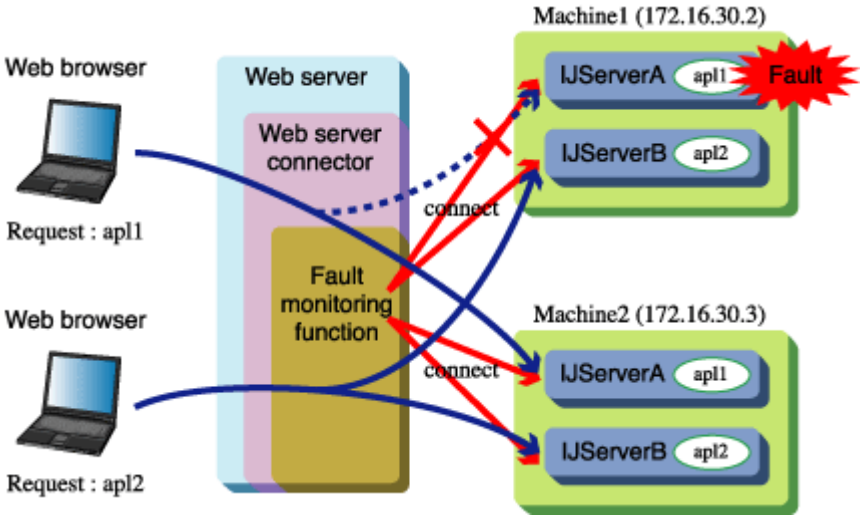


For example, as shown in the figure above, the fault monitoring function issues regular (such as every 60 seconds) pings to the IP addresses of machine 1 and machine 2 and monitors whether there is a response. If the ping response stops because of a fault in the hardware of machine 1, machine 1 is stopped from being distributed automatically, the Web server connector stops the distributing of requests to machine 1.

If a ping response is returned after the machine that failed is repaired, machine 1 is again distributed automatically, and the Web server connector starts the distributing of requests to machine 1 again.

- Port monitoring

  The Servlet container monitors the Servlet container operation status for the TCP port that receives the request, depending on the availability of the TCP connection using connect of socket.



For example, as shown in the figure above, the monitoring function issues regular (such as every 60 seconds) connects to the TCP port that waits for requests from the Servlet container that is opened on machine 1 and machine 2, and monitors whether it is possible to connect. By stopping IJServer A of machine 1, the Servlet container closes the TCP port that is waiting for the request, and TCP connection is no longer possible by connect. Machine 1 is stopped from being distributed automatically, and the Web server connector stops the distributing of requests to the IJServer A Servlet container of machine 1.

If IJServer A is restarted, and TCP connection is made possible again, machine 1 is again distributed automatically, and the Web server connector starts the distributing of requests to the IJServer A Servlet container of machine 1 again.

Advance Preparation

Make the following settings to use the fault monitoring function.

- Operating procedure for separating IJServer and the Web server

  The fault monitoring function can only be used for operation in a configuration in which IJServer and the Web server have been separated. Using a separate Interstage Management Console for IJServer machine and for the Web server machine, click the [System] > [Environment Settings] tabs and specify the Web server and IJServer machines to be separated.

Settings Items

To make the settings for the fault monitoring function, click the [Services] > [Web Server] > [Web Server Connector] > [Fault Monitoring Settings] tabs.

The settings items are as follows:

Item name	Meaning
Monitoring Method	Select one of the following monitoring methods: - No Monitoring The fault monitoring function is not used. - Ping The operation status of the IJServer machine is observed using ping. - Port The operation status of the IJServer machine is observed using ping, operation status of the Servlet container is observed using connect.
Monitoring Interval	Set the interval for monitoring operation status using ping monitoring or port monitoring.
Response Wait Time	Set the wait time for a response after ping or connect has been issued to the IJServer machine. If there is no response, reissue the number of pings or connects that is set as the retry count for a fault, If there are no responses, it can be assumed that there has been a fault.
Retry Count	Set the retry count for when the wait time for a response after ping or connect has been issued to the IJServer machine is exceeded.
Startup Wait Time	Set the startup wait time for the fault monitoring function if it does not start up with the Web server connector. If the fault monitoring function does not start up before this time is exceeded, the Web server connector starts up without the fault monitoring function. If the Web server connector is running without using the fault monitoring function, the Web server connector might distribute requests from the client to a faulty IJServer machine or Servlet container.

Examples of Preparation before Operation

This section contains examples of how to execute the following types of operation.

- Monitoring the operation status of two IJServer machines in a balancing configuration
- Monitoring the operation status of an IJServer machine and a Servlet container in a balancing configuration

Monitoring the Operation status of two IJServer machines in a balancing configuration

[Requirements]

Server machine A (IP Address: 172.16.30.1)	Web server must be running
Server machine B (IP Address: 172.16.30.2)	IJServer must be running
Server machine C (IP Address: 172.16.30.3)	IJServer must be running

[Setting procedure]

1. Using the Interstage Management Console on server machine B, select [WorkUnit] > [Create New] tab and create a WorkUnit with the following settings.

Item Name	**Setting Value Example**
WorkUnit name	MyIJServer
IP address for the Web server	172.16.30.1
IP address for Servlet container	172.16.30.2
Port number	9000
Timeout	480
The number of concurrent processing tasks	64

2. Using the Interstage Management Console on server machine C, select [WorkUnit] > [Create New] tab and create a WorkUnit with the following settings.

Item Name	**Setting Value Example**
WorkUnit name	MyIJServer
IP address for the Web server	172.16.30.1
IP address for Servlet container	172.16.30.3
Port number	9000
Timeout	480
The number of concurrent processing tasks	64

3. Using the Interstage Management Console for server machine A, click the [Services] > [Web Server] > [Web Server Connector] > [Create New] tabs, and create connection information for the following settings:

Item Name	Setting Value Example
WorkUnit name	MyIJServer
IP address for Servlet container : Port number	172.16.30.2:9000 172.16.30.3:9000
Timeout	480
The number of concurrent processing tasks	64

4. Using the Interstage Management Console for server machine A, click the [Services] > [Web Server] > [Web Server Connector] > [Fault Monitoring Settings] tabs, and make the following settings for the fault monitoring function:

Item Name	Setting Value Example
Monitoring Method	ping monitoring
Monitoring Interval	60
Response Wait Time	10
Retry Count	3

5. Using the Interstage Management Console for server machine A, click the [Services] > [Web Server] tabs, and reboot the Web server.

Monitoring the Operation Status of an IJServer Machine and a Servlet Container in a Balancing configuration

Requirements

Server machine A (IP Address: 172.16.30.1)	Web server must be running
Server machine B (IP Address: 172.16.30.2)	IJServer must be running
Server machine C (IP Address: 172.16.30.3)	IJServer must be running

Setting Procedure

1. Using the Interstage Management Console on server machine B, select [WorkUnit] > [Create New] tab and create a WorkUnit with the following settings.

Item Name	Setting Value Example
WorkUnit name	MyIJServer
IP address for the Web server	172.16.30.1
IP address for Servlet container	172.16.30.2
Port number	9000 9001
Timeout	480
The number of concurrent processing tasks	64

2. Using the Interstage Management Console on server machine C, select [WorkUnit] > [Create New] tab and create a WorkUnit with the following settings.

Item Name	Setting Value Example
WorkUnit name	MyIJServer
IP address for the Web server	172.16.30.1
IP address for Servlet container	172.16.30.3
Port number	9000 9001
Timeout	480
The number of concurrent processing tasks	64

3. Using the Interstage Management Console for server machine A, click the [Services] > [Web Server] > [Web Server Connector] > [Create New] tabs, and create connection information for the following settings:

Item Name	Setting Value Example
WorkUnit name	MyIJServer
IP address for Servlet container : Port number	172.16.30.2:9000 172.16.30.2:9001 172.16.30.3:9000 172.16.30.3:9001
Timeout	480
The number of concurrent processing tasks	64

4. Using the Interstage Management Console for server machine A, click the [Services] > [Web Server] > [Web Server Connector] > [Fault Monitoring Settings] tabs, and make the following settings for the fault monitoring function: If port monitoring is selected, all the servers in 'Servlet container IP address : Port number' of step 3 above are monitored.

Item Name	Setting Value Example
Monitoring Method	port monitoring
Monitoring Interval	60
Response Wait Time	10
Retry Count	3

5. Using the Interstage Management Console for server machine A, click the [Services] > [Web Server] tabs, and reboot the Web server.

Viewing the Operation Status

The following command can be used to display the operation status of the distribution destination IJServer in the Web server machine. Refer to Reference Manual (Command Edition) for details of the command.

```
svmondspstat
```

Viewing the Status of all IJServers

Execute the svmondspstat command to view the operation status of all balancing IJServers. There is no need to specify an option.

Example

Conditions:

- Distributing is made to 2 IJServers, the IJServer names of which are MyIJServer1 and MyIJserver2
- The IP address and port number of the distributing destination Servlet container are as follows:
  172.16.30.2:9000
  172.16.30.2:9001
  172.16.30.3:9000
  172.16.30.3:9001
- The fault monitoring method is port monitoring
- Only 172.16.30.2:9001in MyIJServer2 is faulty

If the conditions are as shown above, the following contents are displayed.

```
Status IP Address : Port Number : WorkUnit Name
---------------------------------------
ACTIVE  172.16.30.2:9000:MyIJServer1
DOWN    172.16.30.2:9001:MyIJServer2
ACTIVE  172.16.30.3:9000:MyIJServer1
ACTIVE  172.16.30.3:9001:MyIJServer2
```

Viewing the Status of a Specific IJServer

As shown below, specify the WorkUnit name set in 'WorkUnit name' of the Web server connector and then execute the command to view the operation status of a specific IJServer.

Example

```
svmondspstat –i MyIJServer1
```

Conditions:

- Distributing is made to 2 IJServers, the IJServer names of which are MyIJServer1 and MyIJserver2
- The IP address and port number of the distributing destination Servlet container are as follows:
  172.16.30.2:9000
  172.16.30.2:9001
  172.16.30.3:9000
  172.16.30.3:9001
- The fault monitoring method is port monitoring
- All the Servlet containers in MyIJServer1 are under operation

If the conditions are as shown above, the following contents are displayed.

```
Status IP Address : Port Number : WorkUnit Name
---------------------------------------
ACTIVE  172.16.30.2:9000:MyIJServer1
ACTIVE  172.16.30.3:9000:MyIJServer1
```

Note

Note the following points when using the fault monitoring function:

- All balancing servers that are set using the Web server connector 'Servlet container IP address : Port number' item are monitored according to the conditions set for Fault Monitoring Settings. It is not possible to monitor according to conditions set for a separate IJServer.
- If there is a firewall between the Web server machine and the IJServer machine, settings allowing passage of the PING from the Web server machine IP address to the balancing IJServer machine IP address must be made in the firewall.

- If the ping or connect response is delayed or lost because of high-intensity networking equipment along the route between the Web server machine and the IJServer machine, IJServer might in fact determine that a fault has occurred even if there is operation status. Set appropriate values for 'Response Wait Time' and 'Retry Count' according to the status of the route between the Web server and the IJServer machine.
- The fault monitoring function cannot be used when Traffic Director executes balancing between the Web server machine and the IJServer machine.
- If the fault monitoring function is enabled, or the fault monitoring function settings are changed, the obtaining of the distributing destination status starts at the point when the Web server is started or rebooted.

  Additionally, if a Web server is not rebooted after the settings are changed, the obtaining of the distributing destination status starts when the first access from the client occurs, according to the changed settings.
- Requests might be distributed to the faulty distributing destination immediately after the Web server is started until the fault monitoring function judges that it is faulty.

  The time from the start of the obtaining of the distributing destination status until the judgment that it is faulty is [Response Wait Time x Fault Retry Count].
- If the Web server is not rebooted after the fault monitoring function is enabled, or after the fault monitoring function settings are changed, requests might be distributed to the faulty distributing destination until the fault monitoring function judges that it is faulty, even if the distributing destination is faulty at the point when the first access from the client occurs.

  The time from the start of the obtaining of the distributing destination status until the judgment that it is faulty is [Response Wait Time x Fault Retry Count].
- If a fault does actually occur in the distributing destination, requests might be distributed to the faulty distributing destination until the fault monitoring function judges that it is faulty.

  The time from the actual occurrence of the fault in the distributing destination until the judgment that it is faulty is [Fault monitoring interval + Response Wait Time x Fault monitoring Retry Count].
- If a fault does actually occur in the distributing destination, requests might be distributed to the faulty distributing destination until the fault monitoring function judges that it is faulty.

  In this case, the response to the request is delayed for about 1 minute, or the status code and '500 Internal Server Error' error message are notified from the Web browser.

  In this case, after failure detection is performed, the request can be distributed following the next access.
- Requests are not distributed to the distributing destination that is recovered from the point when the faulty distributing destination is recovered until the fault monitoring function judges that there is a recovery.

  The time from the actual recovery of the distributing destination until the judgment that it has been recovered is [Fault monitoring interval].

Procedure for Using JTS

This section explains the procedure for using JTS.

Flow to Operation Start

To use the application using the distributed transaction function (JTS), the Database Linkage Service is needed. Additionally, the JTS can be used from a Web application, an EJB application or a J2EE application client.

For information on installation and environment setup of each of these services, see the Installation Guide and the Interstage Operator's Guide.

For details on the distributed transaction function and JTA interface distributed by JTS, refer to Part IV, JTS/JTA Edition.

The flow to the operation start of a JTS application, which is assumes an EJB application is operating, is shown below.

1. Setting Resource Manager Environment
2. Setting the Transaction Service Environment
3. Storing Resource Definition Information
4. Starting the Database
5. Starting the Transaction Service
6. Starting the Application

Note

The Interstage Management Console displays 'transaction service (JTS RMP)' for the JTS resource management program.

1. Setting Resource Manager Environment

Refer to the resource manager manuals for details of the resource manager environment setup.

Necessary classpaths must be set to use various JTS resource managers. Refer to the manuals of each resource manager for details of the necessary class libraries.

To set resource manager classpaths, open the system environment setup on the Interstage Management Console and set the classpaths in [J2EE properties].

- Database JDBC driver

  The database JDBC drive must be set when a database is used.

  Example: When an Oracle database is used

  classes12.zip、nls_charset12.zip

- Resource adapter class library

  A resource adapter class library must be set when the connector is used to link with the resource adapter.

Note

If the Interstage Management Console is not to be used, set environment variable classpath as a system environment variable.

Restart Interstage to validate the environment variable.

Notes on using Oracle as a resource manager

- Database construction

  When Oracle8i is to be used with the distributed transaction function using JTS and EJB, the database must be configured in such a way that Oracle IJServer is enabled.

  When Oracle9i is to be used, the database must be configured in such a way that Oracle Enterprise Java Engine (Oracle EJE) is enabled.

  When Oracle9i is to be used with the distributed transaction function using JTS and EJB, the database must be configured in such a way that Oracle Enterprise Java Engine (Oracle EJE) is enabled.

- Database setting

  The following settings are required to enable the use of distributed transactions with an Oracle database.

1. Log on to sqlplus as a SYS user.

```
(Oracle8i)
sqlplus sys/password@ORACLE_SID
(Oracle9i)
sqlplus "sys/password@ORACLE_SID AS SYSDBA"
```

2. Execute the following sql.

```
grant select on DBA_PENDING_TRANSACTIONS to username
* For username, specify the user name that is set in the data source definition.
```

2. Setting the Transaction Service Environment

Using the Interstage Management Console, set the transaction service (OTS) environment.

Note

If the detailed setup is modified, the transaction service environment is reconfigured. The JTS resource definition information stored previously is completely deleted.

To use the target resource again for global transactions, select and apply 'Use for global transactions' in the resource environment setup.

3. Storing Resource Definition Information

Use the Interstage Management Console to store JTS resource definition information.

- To use a JDBC data source for global transactions, select JDBC data source creation from the service list, and select and apply 'Use for global transactions.'
- To use the connector for global transactions, select resource adapter deployment from the service list, and select and apply 'Use for global transactions.'
- To use a resource definition already stored for global transactions, select target resource environment setup from the service list, and select and apply 'Use for global transactions.'

Note

- The database supported by the JDBC data source is only Oracle.

  With Symfoware or SQL Server, 'Use for global transactions' cannot be selected.

- When Oracle is used as a resource manager for distributed transaction processing using JTS, note the following on data source setting:

  If two or more data sources that reference the same database instance on the same host are to be registered, take the following actions:

  1. Define the IP address of the host containing the database in the following file so that it will be a different host name.

  - %SystemRoot%\system32\drivers\etc\hosts

  /etc/hosts

  Example

  123.123.123.110 oracle_db1
  123.123.123.110 oracle_db2

  2. Write the above host name for the connection host name in the JDBC data source definition. Do so for each data source.

  As a result, different database URLs are set for the same host and instance.

Example of server URL

Datasource1 Server URL:jdbc:oracle:thin:@oracle_db1:1521:ora8i
Datasource1 Server URL:jdbc:oracle:thin:@oracle_db2:1521:ora8i

4. Starting the Database

Start the database.

For details on how to start the database, refer to the database manual.

5. Starting the Transaction Service

Start Interstage using the Interstage Management Console. Start also the transaction service.

If the transaction service environment is set in advance during system environment setup, the transaction service is automatically started when Interstage is started.

Note

The transaction service (JTS RMP) performs failure recovery processing in accordance with activation of the linked database and therefore is not automatically started when the server machine is rebooted. For this reason, one of the following operations must be performed after activation of the database:

- Restart Interstage using the Interstage Management Console.
- Enter the otsstartrsc command to start the JTS resource management program.

6. Starting the Application

Start the WorkUnit using the Interstage Management Console.

Flow to Operation End

The following explains the procedure for terminating operation.

1. Stopping the application

   Stop the relevant WorkUnit using the Interstage Management Console.

2. Stopping the transaction service

   Stop Interstage using the Interstage Management Console. The transaction service is automatically stopped.

3. Stopping the database

   Stop the database.

   Refer to the database manual. for details on how to stop the database.

4. Canceling resource definition information

   Using the Interstage Management Console, cancel the resource definition used for global transactions.

5. Canceling the transaction service environment set up

   Using the Interstage Management Console, cancel the transaction service (OTS) environment set up.

Procedure for Using JMS

This section explains the procedure for using JMS.

Flow to Operation Start

To use JMS, the following components are needed. Install ObjectDirector EventService and JMS to perform custom installation. ObjectDirector and J2EE Common are installed as standard features.

For information on custom installation, see Chapter 1, 'Installation' in the Installation Guide.

- ObjectDirector
- ObjectDirector EventService
- J2EE Common
- JMS

The following explains the procedure for starting operation.

1. Environment Settings before Operation of the Event Channel Operation Machine

   Set the environment before operation of the event channel operation machine. Refer to 'Environment Settings for the Event Channel Operation Machine' in Chapter 16 for details.

2. Environment Settings before Operation of the JMS Application Operation Machine

   Set the environment before operation of the JMS application operation machine. Refer to 'Environment Settings for the JMS Application Operation Machine' in Chapter 16 for details.

3. Operation Start of the Event Channel Operation Machine

   1) Starting Interstage

   Activate the Event Service by starting Interstage by using the Interstage Management Console.

   2) Starting the Static Event Channel

   Start the event channel used for sending/receiving messages by the JMS application by using the Interstage Management Console.

   Note:

   If automatic start of the event channel is defined in advance, the event channel is automatically started when Interstage (event service) starts. The setting of automatic start of the event channel can be changed using the Interstage Management Console. The default is 'start automatically.'

4. Operation Start of the JMS Application Operation Machine

   Start the JMS application by using directly the java commands.

Flow to Operation End

The following explains the procedure for terminating operation.

1. Operation End of the JMS Application Operation Machine

   Stop the JMS application.

2. Operation End of the Event Channel Operation Machine

   1) Stopping the Static Event Channel

   Forcibly stop the event channel used for sending/receiving messages by the JMS application by using the Interstage Management Console.

   2) Stopping Interstage

   Stop the Event Service forcibly by stopping Interstage by using the Interstage Management Console.

3. Environment Deletion of the JMS Application Operation Machine

   Delete the environment after operation end of the JMS application operation machine. Refer to 'Environment Settings for the JMS Application Operation Machine' in Chapter 18 for details.

4. Environment Deletion of the Event Channel Operation Machine

   Delete the environment after operation end of the event channel operation machine. Refer to 'Environment Settings for the Event Channel Operation Machine' in Chapter 18 for details.

Monitoring the Operational Status of an Event Channel

The Interstage Management Console can be used to monitor the operational status of the event channel used for sending/receiving messages by the JMS application.

Event channel information is outlined in the following table.

	Display Item	**Description**
1	group	Group name included in an event channel. Dynamic generation: Event Factory (when using TemporaryTopic or TemporaryQueue) Static generation: Event channel name created by the user
2	ChannelName	Event channel name. An ID is used for dynamically generated channels of the Notification Service. (when using TemporaryTopic or TemporaryQueue)
3	Type	Event channel type Used for Topic: Publish/Subscribe messaging model Used for Topic: Queue:Point-To-Point messaging model Used for Topic: TemporaryTopic:Publish/Subscribe messaging model Used for Topic: TemporaryQueue:Point-To-Point messaging model

	Display Item	Description
4	Destination	JNDI name of destination definition associated with an event channel.
5	unit	Unit ID of storage destination used in nonvolatile channel operation mode.
6	Status	Status of the event channels. Active: The event channel is active. Activating: The event channel is being activated. Stopped: The event channel is stopped. Stopping: The event channel is being stopped or it is being stopped in blockade end mode.
7	QueueCount	Number of messages currently stored in the event channels.
8	ConsumerCount	Number of subscribers or receivers that can be connected to the event channels. (Note)
9	SupplierCount	Number of publishers or senders that can be connected to the event channels.

Note) If an EJB Message-driven Bean is used, the following values are added as the number of connected consumers.

- Point-To-Point messaging model

  Windows

  Number of connected consumers = 1

  Linux Solaris OE

  Increase the number of connected consumers from the default value to the maximum value according to the communication status:

  – Number of connected consumers (default) = [Process Concurrency] of the IJServer * [Default start number of instances (number of threads executed simultaneously)] of the Message-driven Bean

  – Number of connected consumers (maximum) = [Process Concurrency] of the IJServer * [Default start number of instances (number of threads executed simultaneously)] of the Message-driven Bean * 2

- Publish/Subscribe messaging model

  Number of connected consumers = 1

Note

The messages stored in the event channel are deleted on one of the following:

- Topic type event channel in persistent operation mode
  - The consumer (subscriber) fetches the messages
  - The message survival time is reached.
  - The consumer (subscriber) connection information is retrieved.
  - The event channel is forcibly stopped.
- Queue type event channel in persistent operation mode
  - The consumer (Receiver) fetches the messages
  - The message survival time is reached.
  - The event channel is forcibly stopped.
- Topic type event channel in non-persistent operation mode
  - The consumer (subscriber) fetches the messages
  - The message survival time is reached.
  - The consumer (subscriber) connection information is retrieved.
- Queue type event channel in non-persistent operation mode
  - The consumer (Receiver) fetches the messages
  - The message survival time is reached.

  * The messages stored in the Queue type event channel used in nonvolatile operation mode cannot be deleted automatically by forcibly stopping the event channel. To delete these messages, use the consumer (receiver) to fetch these messages.

When the event channel is closed, the event channel does not stop as far as messages are stored in the channel. The event channel stops when the stored messages are deleted as the result of distribution to the consumer or expiration of the survival time.

The Interstage Management Console can be used to reference the status of the message save destination (unit).

The status information on the save destination (unit) displayed includes the following:

Unit information is outlined in the following table.

	Display Item	**Description**
1	Unit ID	Unit name
2	Unit Mode	Type of unit used in persistent channel operation mode Standard: Standard unit Extended: Extended unit
3	System utilization (%)	Utilization of system file in the storage directory
4	Event data utilization (%)	Utilization of invent data file in the storage directory
5	Number of system areas	Number of system data storage areas used in persistent channel operation mode
6	Number of event data areas	Number of event data storage areas used in persistent channel operation mode

Procedure for Using JavaMail

This section explains the procedure for using JavaMail.

Before Operation

1. Creating an application

   Create an application used to send or receive mail. For details refer to the following:

   – Mail Sending Application

   – Mail Receiving Application

2. Setting the Mail Server Environment

   To send and receive mail using the JavaMail application, the SMTP server to send mail and the POP3 server or IMAP server to receive mail must be available.

   Set up the environments for the servers to send and receive mail.

   For details of the environment settings of the mail server, refer to the mail server manual.

3. Starting up the Mail Server

   Start up the SMTP server for sending mail and the POP3 server or IMAP server for receiving mail.

   For the method of starting up the server, refer to the mail server manual.

4. Setting the Resource

   Set up the JavaMail resource.

   For the method of setting up the JavaMail resource, refer to Chapter 3, JNDI.

After Operation

1. Stopping the Mail Server

   Stop the mail server. For the method of stopping the server, refer to the mail server manual.

Mail Sending Application

The procedure for Mail sending is shown as follows.

1. Lookup Processing of JavaMail Resources

Create the JavaMail resource lookup processing.

```
// Mail resource lookup processing
      InitialContext nctx = new InitialContext();
      session = (Session) nctx.lookup("java:comp/env/mail/MailSession");
  }
  catch(NamingException ex) { }
```

2. Creating a message

Create a message to be sent.

Set up the following items to the message:

- Sender (From)
- Destination (To)
- Destination (Cc)
- Destination (Bcc)
- Title (Subject)
- Body

```
// Create a message
  MimeMessage msg = null;
  try {
   // Create a message
      msg = new MimeMessage(session);
   // Sender setting (From)
      msg.setFrom(new InternetAddress(“<from-address>”));
   // Destination setting (To)
      Address[] toAddress = {new InternetAddress(“<to-address>”)};
      msg.setRecipients(Message.RecipientType.TO, toAddress);
   // Destination setting (Cc)
      Address[] ccAddress = {new InternetAddress(“<cc-address>”)};
      msg.setRecipients(Message.RecipientType.CC, ccAddress);
   // Destination setting (Bcc)
      Address[] bccAddress = {new InternetAddress(“<bcc-address>”)};
      msg.setRecipients(Message.RecipientType.BCC, bccAddress);
   // Title setting (Subject)
      String subject = new String("<Subject>");
      msg.setSubject(subject);
   // Body setting
      String msgTxt = new String("<Message Text>");
      msg.setText(msgTxt);
  }
  catch(AddressException ex) { }
  catch(MessagingException ex) { }
```

3. Making a Connection with the SMTP Server

Make a connection with the SMTP server

```
// Make a connection with the SMTP server
  Transport transport = null;
  try {
      transport = session.getTransport("smtp");
      transport.connect();
  }
  catch(NoSuchProviderException ex) { }
  catch(MessagingException ex) { }
```

4. Sending the Message

Send the created message.

```
// Send the message
  try {
      transport.send(msg);
  }
  catch(MessagingException ex) { }
```

Mail Receiving Application

The procedure for Mail receiving is shown as follows.

1. Lookup Processing of JavaMail Resources

Look up JavaMail resources.

```
// Look up mail resources
  Session session = null;
  try {
      InitialContext nctx = new InitialContext();
      session = (Session) nctx.lookup("java:comp/env/mail/MailSession");
  }
  catch(NamingException ex) { }
```

2. Making a Connection with the Mail Server

To make a connection with a POP3 server:

```
// make a connection with a POP3 server
  Store store = null;
  try {
      store = session.getStore("POP3");       /*make a connection with a POP3
server */
      store.connect("<hostname>", "<user>", "<password>");
  }
  catch(NoSuchProviderException ex) { }
  catch(MessagingException ex) { }
```

To make a connection with an IMAP server:

```
// Make a connection with the IMAP server
  Store store = null;
  try {
      store = session.getStore("imap");    /* Making a connection with the IMAP
server */
      store.connect("<hostname>", "<user>", "<password>");
  }
  catch(NoSuchProviderException ex) { }
  catch(MessagingException ex) { }
```

3. Opening the Receive Directory

Open the receive directory.

```
// Open the receive directory
  Folder inbox = null;
  try {
      Folder rootFolder = store.getDefaultFolder();
      inbox = rootFolder.getFolder("INBOX");
      inbox.open(Folder.READ_WRITE);
  }
  catch(MessagingException ex) { }
```

4. Extracting Messages

Extract the following items from each received message:

- Sender (From)
- Destination (To)
- Destination (Cc)
- Destination (Bcc)
- Title (Subject)
- Body

```
// Extract a message
  try {
      Message msg = inbox.getMessage(1);
   // Extract the sender (From)
      Address[] fromAddress = msg.getFrom();
   // Extract the destination (To)
      Address[] toAddress = msg.getRecipients(Message.RecipientType.TO);
   // Extract the destination (Cc)
      Address[] ccAddress = msg.getRecipients(Message.RecipientType.CC);
   // Extract the destination (Bcc)
      Address[] bccAddress = msg.getRecipients(Message.RecipientType.BCC);
   // Extract the title (Subject)
      String subject = msg.getSubject();
   // Extract the body
      Object content = msg.getContent();
      String text = content.toString();
  }
  catch(MessagingException ex) { }
  catch(IOException ex) { }
```

Customizing and Checking the Operating Environment

Installing the Interstage Application Server automatically installs default IJServer while no J2EE applications are installed. Default IJServer is installed with name IJServer and can be used not only for the operating environment for sample applications but also for actual operation.

If you want to run a J2EE application immediately, install a sample J2EE application or deploy your J2EE application.

This section explains how to customize the operating environment and how to check it in case a J2EE application fails to run.

Customizing the Operating Environment

Setting the Value of Scale-value

This value need not be customized for normal operation.

According to the number of Interstage clients, set the value of scale-value in the isgendef command.

Setting for using the Fujitsu XML Processor

The container may use the Fujitsu XML processor to analyze deployment descriptor files or name conversion files, or the Fujitsu XML processor may need to be used when a J2EE application uses JAXP(Java API for XML Processing). In this case, perform the following customization..

Note that the Fujitsu XML processor may need to be installed as shown below:

The Fujitsu XML processor is not automatically installed when the Interstage Application Server is installed. Install the Fujitsu XML processor by referring to the Installation Guide.

In custom installation mode, the Fujitsu XML processor may not be installed. If needed, install it by referring to the Installation Guide.

[For Web application and EJB application]

Refer to 'Settings of xml parser' in Chapter 1, and customize.

[For J2EE application Client]

Set the following environment variable before the setting of the 'isj2ee.jar' path.

Windows

Environment variable	Setting Value
CLASSPATH	System drive:\Program Files\Common Files\FujitsuXML\xmlpro.jar System drive:\Program Files\Common Files\FujitsuXML\xmltrans.jar

Solaris OE Linux

Environment Variable	Setting Value
CLASSPATH	/opt/FJSVxmlpc/lib/xmlpro.jar

If JDK1.4 is used, the following must be specified for the parameter of the java command to run the Java application.

- When the DOM interface is used
  - Djavax.xml.parsers.DocumentBuilderFactory=com.fujitsu.xml.tree.DocumentBuilderFactoryImpl
- When the SAX interface is used
  - Djavax.xml.parsers.SAXParserFactory=com.fujitsu.xml.parser.SAXParserFactoryImpl

Tuning the CORBA Service Environment Definition

There is no need for customization for normal operation.

The values specified for statements in the CORBA service environment definition on the machine on which Interstage is installed need to be increased.

Refer to 'CORBA Service Environment Definition' in the Tuning Guide for details of the tuning procedure.

- When a client application is added

Statement	Addition Value
Max_IIOP_resp_con	Number of processes of the client application added

- When the EJB application is added

Statement	Addition Value
Max_processes	Number of processes of the EJB application added
Max_exec_instance	Number of processes of the EJB application added x 16 (initial value of the number of threads specified when an EJB application is deployed)

If a client runs on the machine on which Interstage is installed, another value needs to be added to the CORBA service environment definition. Refer to 'Environment setup for referencing EJB' in Chapter 3, for details of the value to be added.

- When IJServer is added

  Note

  The IJServer referred to here is only the type used when a Web application and an EJB application run on separate JavaVMs or when only an EJB application runs.

  The tuning targets are EJB processes in the above two types of IJServer.

Statement	Addition Value
max_processes	Number of processes of the EJB application added
max_exec_instance	Number of processes of the EJB application added x 64 (the maximum values of number of threads specified when IJServer is created)

Note

When an EJB application is used, the no-communication monitoring function of the CORBA service cannot be used.

Checking the Operating Environment

Setting the Environment Variable

In environment variable CLASSPATH, set the following value if it has not already been set.

Windows

```
C:\Interstage\J2EE\lib\isj2ee.jar
```

Solaris OE Linux

```
/opt/FJSVj2ee/lib/isj2ee.jar
```

Environment Setup of Java

(1) Installing Java

- When the Interstage server package is installed

  JDK1.4 is installed in standard installation mode.
  In custom installation mode, select JDK or JRE and install it.

(2) Setting the environment variable

To operate J2EE applications under Interstage, the Java environment must be set up. When the following values are not set as the environment variable PATH, set up a value as an environment variable PATH.

Windows

```
When applying in JDK1.4 environment:C:\Interstage\JDK14\jre\bin
When applying in JRE1.4 environment:C:\Interstage\JRE14\bin
When applying in JDK1.3 environment:C:\Interstage\JDK13\jre\bin
When applying in JRE1.3 environment:C:\Interstage\JRE13\bin
```

Solaris OE Linux

```
When applying in JDK1.4 environment: /opt/FJSVawjbk/jdk14/jre/bin
When applying in JRE1.4 environment: /opt/FJSVawjbk/jre14/bin
When applying in JDK1.3 environment: /opt/FJSVawjbk/jdk13/jre/bin
When applying in JRE1.3 environment: /opt/FJSVawjbk/jre13/bin
```

Note

Depending on the used shell, 'path' must be used as environment variable instead of 'PATH'. Make the settings according to the environment.

When the version is contained throughout the directory, please read in detail before set up.

To avoid incompatibility problems due to the difference in JVM versions, Fujitsu recommends using the same version of JDK/JRE during development, deployment, and operation. In the case of linked operation of J2EE application clients, Web applications, and EJB applications, Fujitsu also recommends that the same version of JDK/JRE be used in order to avoid incompatibility problems. When using the EJB service, it is necessary to make the corresponding setting in the Java environment setting file.

Setting for using IJServer

To use IJServer, a file containing property information required for applications to run is required.

The property information file (orb.properties) is copied to the directory shown below during Interstage installation.

When Java is installed later or additionally using a Solaris OE or Linux custom installation, the file needs to be copied to the respective directories.

Windows

```
(Property information file C:\Interstage\EJB\etc\orb.properties)
(When applying in JDK)%{JAVA_HOME}%\jre\lib
(When applying in JRE)%{JAVA_HOME}%\lib
```

Solaris OE Linux

```
(Property information file:/opt/FJSVejb/etc/orb.properties)
(When JDK)%{JAVA_HOME}%/jre/lib
(When JRE)%{JAVA_HOME}%/lib
```

In addition, IJServer must be set in the Java environment setup file.

In the following cases, set Java additionally in the Java environment setup file.

- If a Java environment setup is not made during Interstage installation
- If Java is installed later using a Solaris OE or Linux custom installation
- If ava is installed additionally using a Solaris OE or Linux custom installation

The Java environment setup file is allocated to the following directory:

Windows

```
C:\Interstage\J2EE\etc\java_config.txt
```

Solaris OE Linux

```
/opt/FJSVj2ee/etc/java_config.txt
```

The setting format and notes on setting are described below.

Format for Setting

Use the following format for setting:

Java version to be used = Java installation directory

- Java version to be used

  Specify the Java version to be used as follows:

```
When using JDK1.4 : JDK14DIR
When using JRE1.4 : JRE14DIR
When using JDK1.3 : JDK13DIR
When using JRE1.3 : JRE13DIR
```

- Java installation directory

  Specify the absolute path of the Java installation directory.

Example

Windows

Example of statement for using JDK1.4 installed in C:\Interstage\jdk14

```
JDK14DIR = C:\Interstage\jdk14
```

Solaris OE Linux

Example of statement for using JDK1.4 installed in /opt/FJSVawjbk/jdk

```
JDK14DIR = /opt/FJSVawjbk/jdk14
```

Note

Windows

- Do not delete the Java environment setup file or change the contents of the file during operation.
- If JDK is selected during Interstage installation, JRE included in JDK cannot be used as a Java environment. In this case, specify JDK14DIR or JDK13DIR for 'Java version used' and use JDK as the Java environment.

Solaris OE Linux

- Set information in the Java environment setup file with administrator authority.
- Do not delete the Java environment setup file or change the contents of the file during operation.
- If JDK14DIR or JDK13DIR is specified for 'Java version used' in the Java environment setup file, do not set JRE under control of JDK1.4 or JDK1.3 for 'Java installation directory.' Instead, set the directory in which JDK1.4 or JDK1.3 is installed.
- If more than one version of Java is installed, click Interstage Management Console > WorkUnit > WorkUnit Settings, and then select the version of Java to be used for each IJServer. If the Java version is not specified, it is determined according to the following order of priority in the Java environment settings file.

  IJServer created in V7.0

  Priority: Java version

  1 : JDK1.4

  2 : JRE1.4

  3 : JDK1.3

  4 : JRE1.3

  IJServer created in V6.0

  Priority: Java version

  1 : JDK1.3

  2 : JRE1.3

  3 : JDK1.4

  4 : JRE1.4

  For example, in a IJServer created in V6.0, if the 'java' version is not specified, and the Java versions described in the Java environment settings file are JDK1.4 andJDK1.3, the higher settings priority JDK1.3 is used for running IJServer.

Settings for Use of the EJB Service Run Command

Before the EJB service run command can be used, the EJB Java environment settings file and environment variable must be set.

In any of the following cases, add Java to the Java environment settings file.

- There were no Java environment settings when Interstage was installed
- Java was installed after a custom installation in Solaris OE or Linux
- Java is installed as an add-on

The Java environment settings file is created in the following directory.

Windows

```
C:\Interstage\EJB\etc\java_config.txt
```

Solaris OE Linux

```
/opt/FJSVejb/etc/java_config.txt
```

Notes about the settings format and settings are explained below.

Format for Setting

Use the following format for setting:

Java version to be used = Java installation directory

- Java version to be used

  Specify the Java version to be used as follows:

```
When using JDK1.4 : JDK14DIR
When using JRE1.4 : JRE14DIR
When using JDK1.3 : JDK13DIR
When using JRE1.3 : JRE13DIR
```

- Java installation directory

  Specify the absolute path of the Java installation directory.

Example

Windows

Example of statement for using JDK1.4 installed in C:\Interstage\jdk14

```
JDK14DIR = C:\Interstage\jdk14
```

Solaris OE Linux

Example of statement for using JDK1.4 installed in /opt/FJSVawjbk/jdk

```
JDK14DIR = /opt/FJSVawjbk/jdk14
```

Note

Windows

- Do not delete the Java environment setup file or change the contents of the file during operation.
- If JDK is selected during Interstage installation, JRE included in JDK cannot be used as a Java environment. In this case, specify JDK14DIR or JDK13DIR for 'Java version used' and use JDK as the Java environment.

Solaris OE Linux

- Set information in the Java environment setup file with administrator authority.
- Do not delete the Java environment setup file or change the contents of the file during operation.
- If JDK14DIR or JDK13DIR is specified for 'Java version used' in the Java environment setup file, do not set JRE under control of JDK1.4 or JDK1.3 for 'Java installation directory.' Instead, set the directory in which JDK1.4 or JDK1.3 is installed.

Solaris OE

Set the following environment variables.

Java version	Settings
JDK1.3	/opt/FJSVawjbk/jdk13/jre/lib/sparc /opt/FJSVawjbk/jdk13/jre/native_threads /opt/FJSVawjbk/jdk13/jre/lib/sparc/hotspot
JRE1.3	/opt/FJSVawjb/jre13/lib/sparc /opt/FJSVawjb/jre13/lib/sparc/native_threads /opt/FJSVawjb/jre13/lib/sparc/hotspot
JDK1.4	/opt/FJSVawjbk/jdk14/jre/lib/sparc /opt/FJSVawjbk/jdk14/jre/lib/sparc/native_threads /opt/FJSVawjbk/jdk14/jre/lib/sparc/client
JRE1.4	/opt/FJSVawjb/jre14/lib/sparc /opt/FJSVawjb/jre14/lib/sparc/native_threads /opt/FJSVawjb/jre14/lib/sparc/client

If more than one version of Java is described in the Java environment settings file, set the Java version with the highest priority in LD_LIBRARY_PATH. The Java version priority is shown in the table below.

Priority: Java version

1 : JDK1.3

2 : JRE1.3

3 : JDK1.4

4 : JRE1.4

Depending on the used shell, 'path' must be used as environment variable instead of 'PATH'. Make the settings according to the environment.

Debugging Application

Application debug information is output to the IJServer log file.

IJServer Log

The following types of information are output to the IJServer log file. This log file can be used to determine application problems. Refer to 'IJServer file configuration' in Chapter 1, for the log output location.

- Container (container.log)
  - Standard output or standard error output of application
  - Output of log method of ServletContext class
  - EJB container log or Servlet container log
  - EJB container or Servlet container error message
  - EJB snap output
- Container information log (info.log)
  - JavaVM process start information (ARGV, ENV)
  - JavaVM process start error message
  - Thread dump
  - Container log output error message

Debug Method

The following methods are available for debugging applications.

- **Debugging using Snap**

  The snap is used to check the logging information.

- **Debugging using application debug information**

  Debug information output to the standard output or standard error output is checked during application execution.

- **Using the Debugger**

  Using the debugger of APWORKS, application operations can be checked while referencing or changing variables in the program.

- **Automatic thread dump collection**

  A thread dump is automatically collected when an application has caused a timeout or returns no response.

- **Debugging using Java method trace**

  The Java method trace function is used to check the arguments and return values of each method.

This section explains each method individually.

Debugging using Snap

Snap is the log of various types of I-O information as J2EE application debug information during J2EE application execution. It can be used as debug information when J2EE applications are developed.

- The log of various types of I-O information as J2EE application debug information during J2EE application execution
- The user debug information of J2EE application

Note

Note that Snap is available only when an IJServer is started with the Interstage Management Console.

Information Output to Snap

Table 2-1 shows the types of information that are output to Snap.

Table 2-1 Information Output to Snap

Type	Output Information
Method information of EJB application invoked by a Client	Output the following types of method information of the EJB application that is invoked by a Client application: Method invocation information Method return information Method exception information Only when the EJB application meets all the following requirements is it possible to output it. The EJB application is deployed using V6 or later Interstage Management Console. In addition, only when the following method is invoked, the information is output. [Session Bean] Home interface method create remove(handle) remove(primarykey) Remote interface method business method remove LocalHome interface method business method remove Local interface method business method remove

Type	Output Information
	[Entity Bean] Home interface method create remove(handle) remove(primarykey) findByPrimaryKey find <Enumeration type> find <Collection type>, find<Object> ejbHome method Remote interface method business method remove LocalHome interface method create remove(primarykey) findByPrimaryKey find <Enumeration type> Local interface method business method remove Refer to Method Information of EJB Application Invoked by a Client for details of output information, formats, and examples.
EJB application method information	The following types of information are output: Method invocation information Method return information Method exception information For details of output information, formats, and examples, refer to EJB Application Method Information.
javax.transaction.UserTransaction API information	The following types of information are output: Method invocation information Method return information Method exception information For details of output information, formats, and examples, refer to javax.transaction.UserTransaction API Information.

Type	Output Information
Database manipulation statement information (Only when the Entity Bean mode is CMP)	The following types of database manipulation information stored in the Container are output: Database manipulation invocation information Database manipulation return information Database manipulation exception information For details of output information, formats, and examples, refer to Database Manipulation Statement Information.
EJB Container transaction control information	If the transaction type is Container and the transaction attribute is Required or RequiresNew, the following information is output, which is used for a container to invoke an API of javax.transaction.TransactionManager: Transaction start (begin) Transaction completion (commit/rollback) Transaction rollback specification (setRollbackOnly) Transaction suspension or resumption (suspend/resume) Refer to EJB Container Transaction Control Information.
J2EE application user debug information	This outputs the debug information that the J2EE application outputs. Refer to Using Application Debugging Information.

As shown in Table 2-2, the output information varies depending on the output level specified when the IJServer starts.

Table 2-2 Output Level and Information

Output level	Output Information
1	The sequence of the J2EE application method can be checked.
2	In addition to level 1 information, the method execution parameter and return information can be checked. When the Entity Bean mode is CMP, database manipulation information is output and therefore the data flow and database relationships can also be checked.
10	J2EE application user debug information can be checked.
11	In addition to the level 1 information, J2EE application user debug information can be checked.
12	In addition to the level 2 information, J2EE application user debug information can be checked.

Snap Environment Setup

To collect snapshots, specify the output level in the WorkUnit setup of IJServer that collects snapshots.

Table 2-3 Snap Environment Setup

Parameter	Value
Java VM option (Java Command Option)	-DFJSNAP= output-level

Unless the disk is short of free space, all types of Snap information are output without limitation according to the output level.

Notes

- When the rapid invocation function is used, Snap information output by every J2EE application deployed in IJServer is stored in the same file.
- If J2EE applications run in thread multiplex mode, all Snap information is output to the same file. Because Snap is output alternately for each thread, do not run J2EE applications in thread multiplex mode.
- If mass data is used for the return value and parameters of an EJB application method, a memory shortage error may occur. When Snap is used, use small amounts of data.
- If a memory shortage occurs during J2EE application execution, no Snap information is output.
- If an environment variable or its value is invalid (such as a spelling error), the IJServer starts but no Snap is output.

Method Information of EJB Application Invoked by a Client

Method information invoked by a Client is output when each method in the EJB application is invoked, returned, or causes an exception.

Output Formats

Level 1

The output formats at individual output levels are shown below:

- **When a method is invoked**

```
Date       Time         : Client Call   :Bean name     Method name
```

- **When a method returns**

```
Date       Time         : Client Return :Bean name      Method name
```

- **When a method returns with an error**

```
Date      Time    : Client Throw  :Bean name    Method name Exception class
 name: Exception detail character string
```

Level 2

- **When a method is invoked**

```
Date      Time          : Client Call   :Bean name     Method name
     Param              : Parameter information
     TranStatus         : Transaction status
```

- **When a method returns**

```
Date       Time         : Client Return :Bean name    Method name
     ReturnValue        : Return value information
     ObjectField        : Field information
     TranStatus         : Transaction status
```

- **When a method returns with an error**

```
Date      Time         : Client Throw  :Bean name    Method name  Exception
class name: Exception detail character string
     TranStatus         : Transaction status
```

Output Information

Output items and output information are summarized in Table 2-4.

Table 2-4 Output Information of Method Invoked by a Client

Output Item	Output Information	Output Level	
		Level 1	Level 2
Date	The date the method was invoked or returned is indicated in 'day/month/year' format.	O	O
Time	The time the method was invoked or returned is indicated in 'hour: minute: second. millisecond' format.	O	O
Call Return Throw	'Call': Indicates that this information was output when the method was invoked. 'Return': Indicates that this information was output when the method returned. 'Throw': Indicates that this information was output because a method exception occurred.	O	O
Bean name	The name of the EJB application that invoked the method is indicated.	O	O
Method name	The name of an invoked method is indicated.	O	O

Output Item	Output Information	Output Level	
		Level 1	Level 2
Exception class name	The class name of the exception caused by method invocation is indicated. If the caused exception includes a detail character string, it is also indicated.	O	O
Parameter information (Param)	Parameter information (parameter type and value) used for method invocation is indicated in the following format: (Type) parameter or (Type) <Object> If no parameter is used, only the item name is indicated. For the array class and java.util package Hashtable, all stored values are output. When a user object (*1) having a public field is used as a parameter, '<Object>' is added and the ObjectField item is output.	X	O
Return value information (ReturnValue)	Method return value information (return value type and value) is indicated in the following format: (Type) return value or (Type) <Object> In case of void, only the item name is indicated. For the array class and java.util package Hashtable, all stored values are output. When a user object (*1) having a public field is used as a return value, '<Object>' is added and the ObjectField item is output.	X	O

Output Item	Output Information	Output Level	
		Level 1	Level 2
Field information (ObjectField)	Object public field information is indicated in the following format: (Type) field name = field value or (Type) field name = <Object> For the primitive or String type, the type, variable name, and value are output. For other types, the type, variable name, and '<Object>' are output.	X	O

O: Item output at the specified output level

X: Item that is not output

*1: String is excluded.

Level 1

Normal end

```
23/10/2000 09:49:20.159 : Client Call :SampleBean  business
23/10/2000 09:49:21.229 : Client Return :SampleBean business
```

Abnormal end

```
23/10/2000 09:49:20.159 : Client Call :SampleBean business
23/10/2000 09:49:21.229 : Client Throw     :SampleBean business
 java.rmi.RemoteException:SampleBean Internal error
```

Level 2

Normal end

```
23/10/2000 09:49:15.454 : Client Call       :SampleBean  business
Param    : (int)1,
(java.lang.String)"Sample In",
(java.util.Hashtable)["one", "two"]
23/10/2000 09:49:15.514 : Client Return     :SampleBean  business
ReturnValue : (pack.Sample)pack.Sample@abc123<Object>
ObjectField: (int)i = 3,
       (java.lang.String)str = "hello"
```

Abnormal end

```
23/10/2000 09:49:20.159 : Client Call :SampleBean business
Param : (int)1,
(java.lang.String)"Sample In",
(java.util.Hashtable)["one", "two"]
23/10/2000 09:49:21.229 : Client Throw    :SampleBean business
java.rmi.RemoteException:SampleBean Internal error
```

EJB Application Method Information

Method information of an EJB application is output when each method in the EJB application is invoked, returned, or causes an exception.

Output Formats

The output formats at individual output levels are shown below:

Level 1

- **When a method is invoked**

```
Date       Time         : Call   :Bean name     Method name
```

- **When a method returns**

```
Date       Time         : Return :Bean name      Method name
```

- **When a method returns with an error**

```
Date       Time    : Throw  :Bean name    Method name Exception class name:
 Exception detail character string
```

Level 2

- **When a method is invoked**

```
Date      Time         : Call   :Bean name     Method name
     Param             : Parameter information
     TranStatus        : Transaction status
```

- **When a method returns**

```
Date       Time        : Return :Bean name    Method name
     ReturnValue       : Return value information
     ObjectField       : Field information
     TranStatus        : Transaction status
```

- **When a method returns with an error**

```
Date      Time         : Throw  :Bean name    Method name  Exception class
name: Exception detail character string
     TranStatus        : Transaction status
```

Output Information

Output items and output information are summarized in Table 2-5.

Table 2-5 Output Information of EJB Application Method

Output Item	Output Information	Output Level	
		Level 1	Level 2
Date	The date the method was invoked or returned is indicated in 'day/month/year' format.	O	O
Time	The time the method was invoked or returned is indicated in 'hour: minute: second. millisecond' format.	O	O
Call Return Throw	'Call': Indicates that this information was output when the method was invoked. 'Return': Indicates that this information was output when the method returned. 'Throw': Indicates that this information was output because a method exception occurred.	O	O
Bean name	The name of the EJB application that invoked the method is indicated.	O	O
Method name	The name of an invoked method is indicated.	O	O
Exception class name	The class name of the exception caused by method invocation is indicated. If the caused exception includes a detail character string, it is also indicated.	O	O
Parameter information (Param)	Parameter information (parameter type and value) used for method invocation is indicated in the following format: (Type) parameter or (Type) <Object> If no parameter is used, only the item name is indicated. For the array class and java.util package Hashtable, all stored values are output. When a user object (*1) having a public field is used as a parameter, '<Object>' is added and the ObjectField item is output.	X	O

Output Item	Output Information	Output Level	
		Level 1	Level 2
Return value information (ReturnValue)	Method return value information (return value type and value) is indicated in the following format: (Type) return value or (Type) <Object> In case of void, only the item name is indicated. For the array class and java.util package Hashtable, all stored values are output. When a user object (*1) having a public field is used as a return value, '<Object>' is added and the ObjectField item is output.	X	O
Field information (ObjectField)	Object public field information is indicated in the following format: (Type) field name = field value or (Type) field name = <Object> For the primitive or String type, the type, variable name, and value are output. For other types, the type, variable name, and '<Object>' are output.	X	O
Transaction status (TranStatus)	The following information is output: When the output item is Call: Transaction status before method invocation. When the output item is Return or Throw: Transaction status after the end of method execution. This item is output regardless of the use of a transaction.	X	O

O: Item output at the specified output level

X: Item that is not output

*1: String is excluded.

Output Examples

Examples of information output at individual output levels are shown below:

Level 1

Normal end

```
23/10/2000 09:49:15.454 : Call   :SampleBean  business
23/10/2000 09:49:15.514 : Return :SampleBean  business
```

Abnormal end

```
23/10/2000 09:49:20.159 : Call   :SampleBean  business
23/10/2000 09:49:21.229 : Throw  :SampleBean  business  java.rmi.EJBException:
                                  SampleBean Internal error
```

Level 2

Normal end

```
23/10/2000 09:49:15.454 : Call   :SampleBean  business
Param             : (int)1,
                    (java.lang.String)"Sample In",
                    (java.util.Hashtable)["one", "two"]
TranStatus        : STATUS_ACTIVE
23/10/2000 09:49:15.514 : Return :SampleBean  business
ReturnValue       : (pack.Sample)pack.Sample@abc123<Object>
ObjectField       : (int)i = 3,
                    (java.lang.String)str = "hello"
TranStatus        : STATUS_NO_TRANSACTION
```

Abnormal end

```
23/10/2000 09:49:20.159 : Call   :SampleBean  business
Param             : (int) 1,
                    (java.lang.String)"Sample In"
                    (java.util.Hashtable)["one", "two"]
TranStatus        : STATUS_ACTIVE
23/10/2000 09:49:21.229 : Throw  :SampleBean  business  java.rmi.EJBException: SampleBean
Internal error
TranStatus        : STATUS_MARKED_ROLLBACK
```

Note

If an EJB application created by INTERSTAGE V3.0 is used as is, some information may not be output or incorrect information may be output. If this occurs, perform re-deployment with Interstage V6.0 or later. All information will then be output normally.

Method information and items that are not output and items that are output incorrectly are shown in Tables 2-6 and 2-7.

- Method information that is not output

  When an Entity Bean business method is invoked, no item is output.

- Items that are not output

Table 2-6 EJB Application Method Information Items not Output

Item Name	
Param	The item name alone is output with no information.
ObjectField	The item name alone is output with no information.

- Items that are output incorrectly

Table 2-7 EJB Application Method Information Item Output Incorrectly

Item Name	
ReturnValue	This information may not be output with the type specified in the Home/Remote interface definition. All information is output with the type used during execution.

javax.transaction.UserTransaction API Information

javax.transaction.UserTransaction API information is output when the javax.transaction.UserTransaction method is used from an J2EE application.

Output Format

The output formats at individual output levels are shown below.

Level 1

- **When a method is invoked**

```
Date      Time         : Call   :javax.transaction.UserTransaction  Method
name
```

- **When a method returns**

```
Date       Time        : Return :javax.transaction.UserTransaction  Method
name
```

- **When a method returns with an error**

```
Date      Time         : Throw  :javax.transaction.UserTransaction  Method
name  Exception class name: Exception detail character string
```

Level 2

- **When a method is invoked**

```
Date      Time        : Call   : javax.transaction.UserTransaction  Method
 name
Param             : Parameter information
TranStatus        : Transaction status
```

- **When a method returns**

```
Date      Time        : Return : javax.transaction.UserTransaction  Method
name
ReturnValue       : Return value information
TranStatus        : Transaction status
```

- **When a method returns with an error**

```
Date  Time     Throw    : javax.transaction.UserTransaction   Method name
Exception class name  Exception detail character string
TranStatus            :Transaction status
```

Output Information

Output items and output information are summarized in Table 2-8.

Table 2-8 Output Information of javax.transaction.UserTransaction API

Output Item	Output Information	Output Level	
		Level 1	Level 2
Date	The date the method was invoked or returned is indicated in 'day/month/year' format.	O	O
Time	The time the method was invoked or returned is indicated in 'hour: minute: second. millisecond' format.	O	O
Call Return Throw	'Call': Indicates that this information was output when the method was invoked. 'Return': Indicates that this information was output when the method returned. 'Throw': Indicates that this information was output because a method exception occurred.	O	O
Method name	The name of an invoked method is indicated.	O	O
Exception class name	The class name of the exception caused by method invocation is indicated. If the caused exception includes a detail character string, it is also indicated.	O	O

Output Item	Output Information	Output Level	
		Level 1	Level 2
Parameter information (Param)	Parameter information for a method is indicated in the '(type) parameter' format.	X	O
Return value information (ReturnValue)	Method return value information is indicated in the '(type) return value' format.	X	O
Transaction status (TranStatus)	The following information is output: When the output item is Call: Transaction status before method invocation When the output item is Return or Throw: Transaction status after the end of method execution	X	O

O: Item output at the specified output level

X: Item that is not output

Output Examples

Examples of information output at individual output levels are shown below.

Level 1

Normal end

```
18/10/2000 18:02:28.647 : Call   :javax.transaction.UserTransaction  getStatus
18/10/2000 18:02:28.647 : Return :javax.transaction.UserTransaction  getStatus
```

Abnormal end

```
18/10/2000 18:02:28.577 : Call   :javax.transaction.UserTransaction  getStatus
18/10/2000 18:02:28.607 : Throw  :javax.transaction.UserTransaction  getStatus
                                   javax.transaction.SystemException: Internal
 error
```

Level 2

Normal end

```
18/10/2000 18:02:28.647 : Call   :javax.transaction.UserTransaction  getStatus
Param             :
TranStatus        :STATUS_MARKED_ROLLBACK
18/10/2000 18:02:28.647 : Return :javax.transaction.UserTransaction  getStatus
ReturnValue       :(int)1
TranStatus        :STATUS_MARKED_ROLLBACK
```

Abnormal end

```
18/10/2000 18:02:28.577 : Call   :javax.transaction.UserTransaction  getStatus
Param             :
TranStatus        :STATUS_MARKED_ROLLBACK
18/10/2000 18:02:28.607 : Throw  :javax.transaction.UserTransaction  getStatus
javax.transaction.SystemException: Internal error

TranStatus        :STATUS_MARKED_ROLLBACK
```

Database Manipulation Statement Information

Database manipulation statement information is output when the following methods are executed:

- prepareStatement method of java.sql.Connection class.
- executeQuery method of java.sql.PreparedStatement class.
- executeUpdate method of java.sql.PreparedStatement class.

Output Format

Output formats at individual levels are shown below.

Level 1

No information is output.

Level 2

- **When a method is invoked:**

```
Date       Time         : Call   : Class name   Method name
Param               : Parameter information
TranStatus          : Transaction status
```

- **When a method returns:**

```
Date       Time         : Return : Class name   Method name
ReturnValue       : Return value information
TranStatus        : Transaction status
```

- **When a method returns with an error**

```
Date      Time        : Throw  : Class name   Method name   Exception class
name:  Exception detail character string
TranStatus      : Transaction status
```

Output Information

Output items and output information are summarized in Table 2-9.

Table 2-9 Output Information of Database Manipulation Statement

Output Item	Output Information	Output Level	
		Level 1	Level 2
Date	The date the database manipulation statement was invoked or returned is indicated in 'day/month/year' format.	X	O
Time	The time the database manipulation statement was invoked or returned is indicated in 'hour: minute: second. millisecond' format.	X	O
Call Return Throw	'Call': Indicates that this information was output when the database manipulation statement was invoked. 'Return': Indicates that this information was output when the database manipulation statement returned. 'Throw': Indicates that this information was output because a database manipulation statement exception occurred.	X	O
Class name	The class name used in execution of the database manipulation statement is indicated.	X	O
Method name	The method name used in execution of the database manipulation statement is indicated.	X	O
Exception class name	The class name of the exception caused by a database manipulation statement is indicated. If the caused exception includes a detail character string, it is also indicated.	X	O
Parameter information (Param)	The SQL statement used in execution of a database manipulation statement is output in the following format: (Type) parameter	X	O
Return value information (ReturnValue)	The return value of a database manipulation statement (return value type and value) is indicated in the following format: (Type) field name = field value	X	O

Output Item	Output Information	Output Level	
		Level 1	Level 2
Transaction status (TranStatus)	The following information is output: [When the output item is Call] Transaction status when a database manipulation statement is invoked [When the output item is Return] Transaction status when a database manipulation statement returns [When the output item is Throw] Transaction status when a database manipulation statement causes an exception	X	O

O: Item output at the specified output level

X: Item that is not output

Output Examples

Examples of information output at individual output levels are shown below.

Level 1

No information is output.

Level 2

Normal end

```
23/10/2000 09:49:15.514 : Call  :SampleCMP java.sql.Connection prepareStatement
      Param         :(java.lang.String)"INSERT INTO CT2.CATEGORY (C_ID,C_NAME)
 VALUES (?,?)"
      TranStatus        :STATUS_ACTIVE
23/10/2000 09:49:15.524 : Return :SampleCMP  java.sql.Connection
prepareStatement

ReturnValue    :(java.sql.PreparedStatement)java.sql.PreparedStatement@1cb0f4
      TranStatus        :STATUS_ACTIVE
```

Abnormal end

```
23/10/2000 09:49:15.514 : Call  :SampleCMP java.sql.Connection prepareStatement
Param        :(java.lang.String)"INSERT INTO CT2.CATEGORY (C_ID,C_NAME) VALUES
(?,?)"
TranStatus     :STATUS_ACTIVE
23/10/2000 09:49:15.524 : Throw  :SampleCMP  java.sql.Connection
prepareStatement  java.sql.SQLException: SQLState received from backend server
TranStatus     :STATUS_ACTIVE
```

EJB Container Transaction Control Information

If the EJB application transaction type is Container and the transaction attribute is Required or RequiresNew, a EJB container uses a method of the javax.transaction.TransactionManager interface.

The following information is output as container transaction control information, when a EJB container invokes an API of javax.transaction.TransactionManager.

- Transaction start (begin)
- Transaction completion (commit/rollback)
- Transaction rollback specification (setRollbackOnly)
- Transaction suspension or resumption (suspend/resume)

Output Format

Output formats at individual levels are shown below:

Level 1

- **When transaction control starts**

```
Date    Time     : Call : javax.transaction.TransactionManager     Method
name
```

- **When transaction control ends**

```
Date    Time     : Return : javax.transaction.TransactionManager   Method
name
```

- **When a method returns with an error**

```
Date    Time     : Throw : javax.transaction.TransactionManager   Method
name  Exception class name:  Exception detail character string
```

Level 2

- **When transaction control starts**

```
Date    Time     : Call : javax.transaction.TransactionManager     Method
name
Param              : Parameter information
TranStatus       : Transaction status
```

- **When transaction control ends**

```
Date     Time      : Return  :  javax.transaction.TransactionManager   Method
 name
ReturnValue              : Return value information
TranStatus         : Transaction status
```

- **When a method returns with an error**

```
Date     Time      : Throw  :  javax.transaction.TransactionManager   Method
name   Exception class name:   Exception detail character string
               TranStatus         : Transaction status
```

Output Information

Output items and output information are summarized in Table 2-10 below:

Table 2-10 Output Information of Container Transaction Control

Output Item	Output Information	Output Level	
		Level 1	Level 2
Date	The starting date and end date of transaction control are indicated in the 'day/month/year' format.	O	O
Time	The time the transaction control was starts or ends is indicated in the 'hour: minute: second. millisecond' format.	O	O
Call Return Throw	'Call': Indicates that this information was output when the method was invoked. 'Return': Indicates that this information was output when the method returned. 'Throw': Indicates that this information was output because a method exception occurred.	O	O
Method name	The name of an invoked method is indicated.	O	O
Exception class name	The class name of the exception caused by method invocation is indicated. If the caused exception includes a detail character string, it is also indicated.	O	O
Parameter information (Param)	Parameter information for a method is indicated in the '(type) parameter' format.	X	O
Return value information (ReturnValue)	Method return value information is indicated in the '(type) return value' format.	X	O

Output Item	Output Information	Output Level	
		Level 1	Level 2
Transaction status (TranStatus)	The following information is output: [When the output item is Call] Transaction status before method invocation [When the output item is Return or Throw] Transaction status after the end of method execution	X	O

O: Item output at the specified output level

X: Item that is not output

Output Examples

Examples of information output at individual output levels are shown below:

Level 1

Normal end

```
18/10/2000 18:02:28.647 : Call   : javax.transaction.TransactionManager  begin
18/10/2000 18:02:28.647 : Return : javax.transaction.TransactionManager  begin
```

Abnormal end

```
18/10/2000 18:02:28.577 : Call  : : javax.transaction.TransactionManager  begin
18/10/2000 18:02:28.607 : Throw  : javax.transaction.TransactionManager  begin
  javax.transaction.SystemException: Internal error
```

Level 2

Normal end

```
18/10/2000 18:02:28.647 : Call   : javax.transaction.TransactionManager  begin
             Param              :
             TranStatus         : STATUS_NO_TRANSACTION
18/10/2000 18:02:28.647 : Return : javax.transaction.TransactionManager  begin
             ReturnValue        :
             TranStatus         : STATUS_ACTIVE
```

Abnormal end

```
18/10/2000 18:02:28.647 : Call   : javax.transaction.TransactionManager  begin
            Param             :
            TranStatus        : STATUS_NO_TRANSACTION
18/10/2000 18:02:28.647 : Throw : javax.transaction.TransactionManager  begin
javax.transaction.SystemException: Internal error
            TranStatus        : STATUS_NO_TRANSACTION
```

J2EE Application User Debug Information

User debug information on the J2EE application is output when the log output method is invoked from the J2EE application.

For details of writing user debug information on each method to a log file, refer to Log Output Method for Support. This chapter explains the procedure of writing user debug information to a snap file, the output form, and the content of the output.

Procedure for Outputting User Debug Information

The procedure of writing user debug information on the J2EE application to a snap file is described in the J2EE application class. The description procedure is as follows.

1. Add the import line.
2. Get the class to output debug information.
3. Call the method of outputting various logs by using the class mentioned in point 2, and output debug information for any J2EE application.

Example

```
package  sample.ejb.entity.bmp;

   import java.rmi.*;
   import javax.ejb.*;
   import java.sql.*;
   import javax.sql.*;
   import javax.naming.*;
   import com.fujitsu.interstage.ejb.container.common.*; // Add logging class ----1

   public class EntityBMPUseLogger implements javax.ejb.EntityBean {
     // Get the class (Logger) to output the debug information
    private static Logger logger = Logger.getLogger("UseLoggerCMP");  // -----------2
    private  boolean isDebug = false;

   ...

    public void setEntityContext(javax.ejb.EntityContext ctx)
      throws javax.ejb.EJBException, java.rmi.RemoteException {
      ...

    public EntityBMPUseLoggerPrimaryKey ejbCreate(int No, String Name, int
Stock)
      throws javax.ejb.DuplicateKeyException, javax.ejbCreateException,
```

```
      javax.ejb.RemoteException {
        if(isDebug) {  // Debug Mode ?
        // Output the snap log
        logger.log (Level.FINE,"ejbCreate START"); // ----------------------------3
        }

      ...
```

Differences in Coding Method by JDK Version

java.util.Logging package is used in JDK1.4. API provided by Interstage is used in JDK1.3 for debugging. The J2EE application created in JDK1.3 and JDK1.4 differs in the following points:

- The import statement is different

JDK Version	Import Statement
JDK1.3	com.fujitsu.interstage.ejb.container.common
JDK1.4	java.util.Logging

- Classpath setting when you compile (for JDK1.3)

  Add the following file to the Classpath when you compile the J2EE application

  Apworks installation directory\ejb\lib\fjcontainer72.jar

Output Format

The date and time etc is added to the user debug information log automatically.

The structure of the log file depends on the chosen output method. The form is decided according to the following.

The output form is different according to API used. Refer to Log Output Method for Support for details.

1. Output only message (Specified string)

```
Date Time : Log Message Log level Message
```

2. Message (Specified string)+ Parameter (Any Object)

```
Date Time: Log Message Log level Message
Log Param:Parameter information
```

3. Message (Specified string)+ Exception (Any Exception)

```
Date Time: Log Message Log level Message Exception information
```

Output Information

Output items and output information are summarized in the table below:

Table 2-11 Output Information of User Debug on EJB Application

Output Item	Output Information
Date	The date when the debug information is output is shown in the 'day/month/year' format.
Time	The time when the debug information is output is shown in the 'time:minutes:second.millisecond' format.
Log Message	Debug information is shown
Log Exception	Debug information (Exception information) is shown.
Log Level	The level of debug information (level specified for the log output method) is output by the following character string. '[FINEST]', '[FINER]', '[FINE]', '[CONFIG]', '[INFO]', '[WARNING]', [SEVERE]'
Message	Debug information (Any character string) that the J2EE application specified is output.
Parameter information (Param)	Parameter information (parameter type and value) used for method invocation is indicated in the following format: (Type) parameter or (Type) <Object> If no parameter is used, only the item name is indicated. For the array class and java.util package Hashtable, all stored values are output. When a user object (*1) having a public field is used as a parameter, '<Object>' is added and the ObjectField item is output.
Exception information	Output exception information (any exception) specified by J2EE application Additionally, when a detailed character string is included in the generated exception, the detailed character string is output as well.
Field information (ObjectField)	Object public field information is indicated in the following format: (Type) field name = field value or (Type) field name = <Object> For the primitive or String type, the type, variable name, and value are output. For other types, the type, variable name, and '<Object>' are output.

Output Examples

Examples of information output at individual output levels are shown below:

1. Output only message (Specified string)

In case where the method logger.log(Level.INFO,'DBAccess start!!') is used:

```
23/10/2000 09:49:15.454 : Log Message: [INFO] DBAccess start!!
```

2. Message (Specified string)+ Parameter (Any Object)

In case where the method logger.log(Level.INFO,' prepareStatement ',sql) is used:

('sql' is java.lang.String type)

```
23/10/2000 09:49:15.454 : Log Message: [INFO] prepareStatement
Log Param : (java.lang.String)”SELECT * FROM EMP_EJB1 WHERE (ID=?)”
```

3. Message (Specified string)+ Exception (Any Exception)

In case where the method logger.log(Level.SEVERE'Error!!', ex) is used:

('ex' is java.lang.Throwable type)

```
23/10/2000 09:49:15.454:LogException:[SEVERE] Error!!
java.lang.NullPointerException
```

Notes

- There is no need to describe any description in the program to output to the snap file, except for getting the user debug information on the J2EE application. There is a possibility that the performance deteriorates by the overhead when the log output method is called. However, if the environment of the snap is not set, or output snap with the output level of the snap is set to 1 or 2 when the IJServer is activated, debug information on the J2EE application is never sent to a snap file. For this reason, take care when programming.
- Do not use methods of the java.util.logging.LogManager class from the J2EE application. There is a possibility of abnormal operating when a snap function is used.

Log Output Method for Support

The following is a list of the log output method for support. For details, refer to Java 2 Platform API Specification.

Table 2-12 Log Output Method for Support

Support Method	Output Debug Information
config(String msg)	Log level = '[CONFIG]' Message = String specified for msg
fine(String msg)	Log level = '[FINE]' Message = String specified for msg
finer(String msg)	Log level = '[FINER]' Message = String specified for msg
finest(String msg)	Log level = '[FINEST]' Message = String specified for msg
info(String msg)	Log level = '[INFO]' Message = String specified for msg
severe(String msg)	Log level = '[SEVERE]' Message = String specified for msg
warning(String msg)	Log level = '[WARNING]' Message = String specified for msg
log(Level level, String msg)	Log level = String for the level Message = String specified for msg
log(Level level, String msg, Object param1)	Log level = String for the level Message = String specified for msg Parameter = String specified for param1
log(Level level, String msg, Object[] params)	Log level = String for the level Message = String specified for msg Parameter = String specified for params
log(Level level, String msg, Throwable thrown)	Log level = String for the level Message = String specified for msg Exception information = information specified for thrown

Support Method	Output Debug Information
log(LogRecord record)	Output the following information that is set in LogRecord Log level =String that is set in setLevel() method. Message = String that is set in setMessage() method Parameter information = String that is set in setParameters()method Exception information = String that is set in setThrown() method If the both parameter and exception information are set at the same time, only exception information is output and parameter information is not output.
Logp (Level level,String sourceClass, String sourceMethod, String msg) .	Log level = String for the level Message = String specified for msg
logp(Level level, String sourceClass, String sourceMethod,String msg,Object param1)	Log level = String for the level Message = String specified for msg Parameter = String specified for param1
logp(Level level, String sourceClass, String sourceMethod, String msg, Object[] params)	Log level = String for the level Message = String specified for msg Parameter = String specified for params
logp(Level level, String sourceClass, String sourceMethod, String msg, Throwable thrown)	Log level = '[FINE]' Message = String specified for msg Exception information = information specified for thrown
logrb(Level level, String sourceClass, String sourceMethod, String bundleName, String msg)	Log level = String for the level Message = String specified for msg
logrb(Level level, String sourceClass, String sourceMethod, String bundleName, String msg, Object param1)	Log level = String for the level Message = String specified for msg Parameter = String specified for param1

Support Method	Output Debug Information
logrb(Level level, String sourceClass, String sourceMethod, String bundleName, String msg, Object[] params)	Log level = String for the level Message = String specified for msg Parameter = String specified for params
logrb(Level level, String sourceClass, String sourceMethod, String bundleName, String msg, Throwable thrown)	Log level = String for the level Message = String specified for msg' Exception information = information specified for thrown
throwing(String sourceClass, String sourceMethod, Throwable thrown)	Log level = '[FINER]' Message = 'THROW' Exception information = information specified for thrown

Snap File Output Example

```
10/12/2001 13:58:57.708 : Call   :JcccCmpEntityUserExRW  setEntityContext
      Param          :(javax.ejb.EntityContext)javax.ejb.EntityContext@1049d3
      TranStatus        :STATUS_NO_TRANSACTION
10/12/2001 13:58:57.708 : Return   :JcccCmpEntityUserExRW  setEntityContext
      ReturnValue       :
      TranStatus        :STATUS_NO_TRANSACTION
10/12/2001 13:58:57.708 : Call   :JcccCmpEntityUserExRW  setEntityContext
      Param          :(javax.ejb.EntityContext)javax.ejb.EntityContext@4a7a12
      TranStatus        :STATUS_NO_TRANSACTION
10/12/2001 13:58:57.708 : Return   :JcccCmpEntityUserExRW  setEntityContext
      ReturnValue       :
      TranStatus        :STATUS_NO_TRANSACTION
10/12/2001 13:58:57.708 : Call   :JcccCmpEntityUserExRW  setEntityContext
      Param          :(javax.ejb.EntityContext)javax.ejb.EntityContext@1444d1
      TranStatus        :STATUS_NO_TRANSACTION
10/12/2001 13:58:57.708 : Return   :JcccCmpEntityUserExRW  setEntityContext
      ReturnValue       :
      TranStatus        :STATUS_NO_TRANSACTION
10/12/2001 13:58:57.708 : Call   :JcccCmpEntityUserExRW  setEntityContext
      Param          :(javax.ejb.EntityContext)javax.ejb.EntityContext@32060c
      TranStatus        :STATUS_NO_TRANSACTION
10/12/2001 13:58:57.708 : Return   :JcccCmpEntityUserExRW  setEntityContext
      ReturnValue       :
      TranStatus        :STATUS_NO_TRANSACTION
10/12/2001 13:58:57.708 : Call   :JcccCmpEntityUserExRW  setEntityContext
      Param          :(javax.ejb.EntityContext)javax.ejb.EntityContext@2b323e
      TranStatus        :STATUS_NO_TRANSACTION
10/12/2001 13:58:57.718 : Return   :JcccCmpEntityUserExRW  setEntityContext
      ReturnValue       :
      TranStatus        :STATUS_NO_TRANSACTION
10/12/2001 13:58:57.718 : Call   :JcccCmpEntityUserExRW  setEntityContext
```

```
      Param          :(javax.ejb.EntityContext)javax.ejb.EntityContext@6fb4be
      TranStatus        :STATUS_NO_TRANSACTION
10/12/2001 13:58:57.718 : Return   :JcccCmpEntityUserExRW  setEntityContext
      ReturnValue       :
      TranStatus        :STATUS_NO_TRANSACTION
10/12/2001 13:58:57.718 : Call   :JcccCmpEntityUserExRW  setEntityContext
      Param          :(javax.ejb.EntityContext)javax.ejb.EntityContext@435a3a
      TranStatus        :STATUS_NO_TRANSACTION
10/12/2001 13:58:57.718 : Return   :JcccCmpEntityUserExRW  setEntityContext
      ReturnValue       :
      TranStatus        :STATUS_NO_TRANSACTION
10/12/2001 13:58:57.718 : Call   :JcccCmpEntityUserExRW  setEntityContext
      Param          :(javax.ejb.EntityContext)javax.ejb.EntityContext@58c528
      TranStatus        :STATUS_NO_TRANSACTION
10/12/2001 13:58:57.718 : Return   :JcccCmpEntityUserExRW  setEntityContext
      ReturnValue       :
      TranStatus        :STATUS_NO_TRANSACTION
10/12/2001 13:58:57.718 : Call   :JcccCmpEntityUserExRW  setEntityContext
      Param          :(javax.ejb.EntityContext)javax.ejb.EntityContext@77eaf8
      TranStatus        :STATUS_NO_TRANSACTION
10/12/2001 13:58:57.718 : Return   :JcccCmpEntityUserExRW  setEntityContext
      ReturnValue       :
      TranStatus        :STATUS_NO_TRANSACTION
10/12/2001 13:58:57.718 : Call   :JcccCmpEntityUserExRW  setEntityContext
      Param          :(javax.ejb.EntityContext)javax.ejb.EntityContext@635bb7
      TranStatus        :STATUS_NO_TRANSACTION
10/12/2001 13:58:57.718 : Return   :JcccCmpEntityUserExRW  setEntityContext
      ReturnValue       :
      TranStatus        :STATUS_NO_TRANSACTION
10/12/2001 13:58:57.718 : Call   :JcccCmpEntityRW  setEntityContext
      Param          :(javax.ejb.EntityContext)javax.ejb.EntityContext@1a8a68
      TranStatus        :STATUS_NO_TRANSACTION
10/12/2001 13:58:57.718 : Return   :JcccCmpEntityRW  setEntityContext
      ReturnValue       :
      TranStatus        :STATUS_NO_TRANSACTION
10/12/2001 13:58:57.728 : Call   :JcccCmpEntityRW  setEntityContext
      Param          :(javax.ejb.EntityContext)javax.ejb.EntityContext@74e571
      TranStatus        :STATUS_NO_TRANSACTION
10/12/2001 13:58:57.728 : Return   :JcccCmpEntityRW  setEntityContext
      ReturnValue       :
      TranStatus        :STATUS_NO_TRANSACTION
10/12/2001 13:58:57.728 : Call   :JcccCmpEntityRW  setEntityContext
      Param           :(javax.ejb.EntityContext)javax.ejb.EntityContext@38de7
      TranStatus        :STATUS_NO_TRANSACTION
10/12/2001 13:58:57.728 : Return   :JcccCmpEntityRW  setEntityContext
      ReturnValue       :
      TranStatus        :STATUS_NO_TRANSACTION
10/12/2001 13:58:57.728 : Call   :JcccCmpEntityRW  setEntityContext
      Param          :(javax.ejb.EntityContext)javax.ejb.EntityContext@5976c2
      TranStatus        :STATUS_NO_TRANSACTION
10/12/2001 13:58:57.728 : Return   :JcccCmpEntityRW  setEntityContext
      ReturnValue       :
      TranStatus        :STATUS_NO_TRANSACTION
10/12/2001 13:58:57.728 : Call   :JcccCmpEntityRW  setEntityContext
```

```
      Param           :(javax.ejb.EntityContext)javax.ejb.EntityContext@3e7de
      TranStatus        :STATUS_NO_TRANSACTION
10/12/2001 13:58:57.728 : Return   :JcccCmpEntityRW  setEntityContext
      ReturnValue       :
      TranStatus        :STATUS_NO_TRANSACTION
10/12/2001 13:58:57.728 : Call   :JcccCmpEntityRW  setEntityContext
      Param          :(javax.ejb.EntityContext)javax.ejb.EntityContext@6bcdbb
      TranStatus        :STATUS_NO_TRANSACTION
10/12/2001 13:58:57.728 : Return   :JcccCmpEntityRW  setEntityContext
      ReturnValue       :
      TranStatus        :STATUS_NO_TRANSACTION
10/12/2001 13:58:57.728 : Call   :JcccCmpEntityRW  setEntityContext
      Param           :(javax.ejb.EntityContext)javax.ejb.EntityContext@fe2b9
      TranStatus        :STATUS_NO_TRANSACTION
10/12/2001 13:58:57.728 : Return   :JcccCmpEntityRW  setEntityContext
      ReturnValue       :
      TranStatus        :STATUS_NO_TRANSACTION
10/12/2001 13:58:57.728 : Call   :JcccCmpEntityRW  setEntityContext
      Param          :(javax.ejb.EntityContext)javax.ejb.EntityContext@6e148b
      TranStatus        :STATUS_NO_TRANSACTION
10/12/2001 13:58:57.728 : Return   :JcccCmpEntityRW  setEntityContext
      ReturnValue       :
      TranStatus        :STATUS_NO_TRANSACTION
10/12/2001 13:58:57.738 : Call   :JcccCmpEntityRW  setEntityContext
      Param           :(javax.ejb.EntityContext)javax.ejb.EntityContext@6d484
      TranStatus        :STATUS_NO_TRANSACTION
10/12/2001 13:58:57.738 : Return   :JcccCmpEntityRW  setEntityContext
      ReturnValue       :
      TranStatus        :STATUS_NO_TRANSACTION
10/12/2001 13:58:57.738 : Call   :JcccCmpEntityRW  setEntityContext
      Param          :(javax.ejb.EntityContext)javax.ejb.EntityContext@6a48be
      TranStatus        :STATUS_NO_TRANSACTION
10/12/2001 13:58:57.738 : Return   :JcccCmpEntityRW  setEntityContext
      ReturnValue       :
      TranStatus        :STATUS_NO_TRANSACTION
10/12/2001 13:58:57.738 : Call   :JcccCmpEntitySysExRW  setEntityContext
      Param          :(javax.ejb.EntityContext)javax.ejb.EntityContext@4dd758
      TranStatus        :STATUS_NO_TRANSACTION
10/12/2001 13:58:57.738 : Return   :JcccCmpEntitySysExRW  setEntityContext
      ReturnValue       :
      TranStatus        :STATUS_NO_TRANSACTION
10/12/2001 13:58:57.738 : Call   :JcccCmpEntitySysExRW  setEntityContext
      Param          :(javax.ejb.EntityContext)javax.ejb.EntityContext@74d93a
      TranStatus        :STATUS_NO_TRANSACTION
10/12/2001 13:58:57.738 : Return   :JcccCmpEntitySysExRW  setEntityContext
      ReturnValue       :
      TranStatus        :STATUS_NO_TRANSACTION
10/12/2001 13:58:57.738 : Call   :JcccCmpEntitySysExRW  setEntityContext
      Param          :(javax.ejb.EntityContext)javax.ejb.EntityContext@61a907
      TranStatus        :STATUS_NO_TRANSACTION
10/12/2001 13:58:57.738 : Return   :JcccCmpEntitySysExRW  setEntityContext
      ReturnValue       :
      TranStatus        :STATUS_NO_TRANSACTION
10/12/2001 13:58:57.738 : Call   :JcccCmpEntitySysExRW  setEntityContext
```

```
      Param          :(javax.ejb.EntityContext)javax.ejb.EntityContext@20225b
      TranStatus        :STATUS_NO_TRANSACTION
10/12/2001 13:58:57.738 : Return   :JcccCmpEntitySysExRW  setEntityContext
      ReturnValue       :
      TranStatus        :STATUS_NO_TRANSACTION
10/12/2001 13:58:57.738 : Call   :JcccCmpEntitySysExRW  setEntityContext
      Param          :(javax.ejb.EntityContext)javax.ejb.EntityContext@2f8b5a
      TranStatus        :STATUS_NO_TRANSACTION
10/12/2001 13:58:57.758 : Return   :JcccCmpEntitySysExRW  setEntityContext
      ReturnValue       :
      TranStatus        :STATUS_NO_TRANSACTION
10/12/2001 13:58:57.758 : Call   :JcccCmpEntitySysExRW  setEntityContext
      Param          :(javax.ejb.EntityContext)javax.ejb.EntityContext@78bc3b
      TranStatus        :STATUS_NO_TRANSACTION
10/12/2001 13:58:57.758 : Return   :JcccCmpEntitySysExRW  setEntityContext
      ReturnValue       :
      TranStatus        :STATUS_NO_TRANSACTION
10/12/2001 13:58:57.758 : Call   :JcccCmpEntitySysExRW  setEntityContext
      Param           :(javax.ejb.EntityContext)javax.ejb.EntityContext@ddc4c
      TranStatus        :STATUS_NO_TRANSACTION
10/12/2001 13:58:57.758 : Return   :JcccCmpEntitySysExRW  setEntityContext
      ReturnValue       :
      TranStatus        :STATUS_NO_TRANSACTION
10/12/2001 13:58:57.768 : Call   :JcccCmpEntitySysExRW  setEntityContext
      Param          :(javax.ejb.EntityContext)javax.ejb.EntityContext@7a1767
      TranStatus        :STATUS_NO_TRANSACTION
10/12/2001 13:58:57.768 : Return   :JcccCmpEntitySysExRW  setEntityContext
      ReturnValue       :
      TranStatus        :STATUS_NO_TRANSACTION
10/12/2001 13:58:57.768 : Call   :JcccCmpEntitySysExRW  setEntityContext
      Param          :(javax.ejb.EntityContext)javax.ejb.EntityContext@3e41ec
      TranStatus        :STATUS_NO_TRANSACTION
10/12/2001 13:58:57.768 : Return   :JcccCmpEntitySysExRW  setEntityContext
      ReturnValue       :
      TranStatus        :STATUS_NO_TRANSACTION
10/12/2001 13:58:57.768 : Call   :JcccCmpEntitySysExRW  setEntityContext
      Param          :(javax.ejb.EntityContext)javax.ejb.EntityContext@5a2cef
      TranStatus        :STATUS_NO_TRANSACTION
10/12/2001 13:58:57.768 : Return   :JcccCmpEntitySysExRW  setEntityContext
      ReturnValue       :
      TranStatus        :STATUS_NO_TRANSACTION
10/12/2001 13:59:17.166 : Call   :JcccCmpCBSRW  setSessionContext
      Param        :(javax.ejb.SessionContext)javax.ejb.SessionContext@516fc1
      TranStatus        :STATUS_NO_TRANSACTION
10/12/2001 13:59:17.196 : Return   :JcccCmpCBSRW  setSessionContext
      ReturnValue       :
      TranStatus        :STATUS_NO_TRANSACTION
10/12/2001 13:59:17.296 : Call   :JcccCmpCBSRW  ejbCreate
      Param             :(int)1
      TranStatus        :STATUS_NO_TRANSACTION
10/12/2001 13:59:17.306 : Call   :JcccCmpCBMRW  setSessionContext
      Param        :(javax.ejb.SessionContext)javax.ejb.SessionContext@7cd37a
      TranStatus        :STATUS_NO_TRANSACTION
10/12/2001 13:59:17.306 : Return   :JcccCmpCBMRW  setSessionContext
```

```
      ReturnValue       :
      TranStatus        :STATUS_NO_TRANSACTION
10/12/2001 13:59:17.306 : Call   :JcccCmpCBMRW  ejbCreate
      Param             :(int)1
      TranStatus        :STATUS_NO_TRANSACTION
10/12/2001 13:59:17.306 : Return   :JcccCmpCBMRW  ejbCreate
      ReturnValue       :
      TranStatus        :STATUS_NO_TRANSACTION
10/12/2001 13:59:17.306 : Return   :JcccCmpCBSRW  ejbCreate
      ReturnValue       :
      TranStatus        :STATUS_NO_TRANSACTION
10/12/2001 13:59:17.547 : Call   :JcccCmpCBSRW  empUpdate
      Param             :(java.lang.String)"DUMMY",
                         (long)0,

(com.fujitsu.jccc.view.EmployeeEV)com.fujitsu.jccc.view.EmployeeEV@7e845a<Ob
ject>,
                         (boolean)false,
                         (int)1
      ObjectField       :(java.lang.String)id = "9016",
                         (java.lang.String)name = "Dept1",
                         (java.lang.String)dept = "Name1",
                         (int)age = 11,
                         (int)exRes = 99
      TranStatus        :STATUS_NO_TRANSACTION
10/12/2001 13:59:17.547 : Call   :javax.transaction.UserTransaction  begin
      Param             :
      TranStatus        :STATUS_NO_TRANSACTION
10/12/2001 13:59:17.547 : Return   :javax.transaction.UserTransaction  begin
      ReturnValue       :
      TranStatus        :STATUS_ACTIVE
10/12/2001 13:59:17.547 : Call   :JcccCmpCBMRW  afterBegin
      Param             :
      TranStatus        :STATUS_ACTIVE
10/12/2001 13:59:17.547 : Return   :JcccCmpCBMRW  afterBegin
      ReturnValue       :
      TranStatus        :STATUS_ACTIVE
10/12/2001 13:59:17.557 : Call   :JcccCmpCBMRW  empUpdate
      Param             :(java.lang.String)"DUMMY",
                         (long)0,

(com.fujitsu.jccc.view.EmployeeEV)com.fujitsu.jccc.view.EmployeeEV@7e845a<Ob
ject>,
                         (int)1
      ObjectField       :(java.lang.String)id = "9016",
                         (java.lang.String)name = "Dept1",
                         (java.lang.String)dept = "Name1",
                         (int)age = 11,
                         (int)exRes = 99
      TranStatus        :STATUS_ACTIVE
10/12/2001 13:59:17.587 : Call   :JcccCmpEntityUserExRW  ejbActivate
      Param             :
      TranStatus        :STATUS_ACTIVE
10/12/2001 13:59:17.587 : Return   :JcccCmpEntityUserExRW  ejbActivate
```

```
      ReturnValue       :
      TranStatus        :STATUS_ACTIVE
10/12/2001 13:59:19.770 : Call   :JcccCmpEntityUserExRW  java.sql.Connection
prepareStatement
      Param             :(java.lang.String)"SELECT ID,NAME,DEPT,AGE FROM
 SCOTT.EMP_EJB1 WHERE ID = ?"
      TranStatus        :STATUS_ACTIVE
10/12/2001 13:59:19.840 : Return   :JcccCmpEntityUserExRW  java.sql.Connection
prepareStatement

ReturnValue     :(java.sql.PreparedStatement)java.sql.PreparedStatement@e4245
      TranStatus        :STATUS_ACTIVE
10/12/2001 13:59:19.840 : Call   :JcccCmpEntityUserExRW
java.sql.PreparedStatement executeQuery
      Param             :
      TranStatus        :STATUS_ACTIVE
10/12/2001 13:59:19.850 : Return   :JcccCmpEntityUserExRW
java.sql.PreparedStatement executeQuery

ReturnValue       :(oracle.jdbc.driver.OracleResultSetImpl)oracle.jdbc.drive
r.OracleResultSetImpl@20fa83
      TranStatus        :STATUS_ACTIVE
10/12/2001 13:59:19.900 : Call   :JcccCmpEntityUserExRW  ejbLoad
      Param             :
      TranStatus        :STATUS_ACTIVE
10/12/2001 13:59:19.900 : Return   :JcccCmpEntityUserExRW  ejbLoad
      ReturnValue       :
      TranStatus        :STATUS_ACTIVE
10/12/2001 13:59:19.920 : Call   :JcccCmpEntityUserExRW  fsetNAME
      Param             :(java.lang.String)"Dept1"
      TranStatus        :STATUS_ACTIVE
10/12/2001 13:59:19.930 : Throw   :JcccCmpEntityUserExRW  fsetNAME
com.fujitsu.jccc.cmp.entity.UserException: JcccBmpEtyEx#bussiness : throw
UserException for UserException-Test
      TranStatus        :STATUS_ACTIVE
10/12/2001 13:59:19.930 : Return   :JcccCmpCBMRW  empUpdate
      ReturnValue       :(int)1
      TranStatus        :STATUS_ACTIVE
10/12/2001 13:59:19.930 : Call   :javax.transaction.UserTransaction  getStatus
      Param             :
      TranStatus        :STATUS_ACTIVE
10/12/2001 13:59:19.930 : Return  :javax.transaction.UserTransaction getStatus
      ReturnValue       :(int)0
      TranStatus        :STATUS_ACTIVE
10/12/2001 13:59:19.930 : Call   :javax.transaction.UserTransaction  commit
      Param             :
      TranStatus        :STATUS_ACTIVE
10/12/2001 13:59:19.930 : Call   :JcccCmpCBMRW  beforeCompletion
      Param             :
      TranStatus        :STATUS_ACTIVE
10/12/2001 13:59:19.930 : Return   :JcccCmpCBMRW  beforeCompletion
      ReturnValue       :
      TranStatus        :STATUS_ACTIVE
10/12/2001 13:59:19.930 : Call   :JcccCmpEntityUserExRW  ejbStore
```

```
      Param             :
      TranStatus        :STATUS_ACTIVE
10/12/2001 13:59:19.930 : Return   :JcccCmpEntityUserExRW  ejbStore
      ReturnValue       :
      TranStatus        :STATUS_ACTIVE
10/12/2001 13:59:19.930 : Call   :JcccCmpEntityUserExRW  ejbPassivate
      Param             :
      TranStatus        :STATUS_ACTIVE
10/12/2001 13:59:19.930 : Return   :JcccCmpEntityUserExRW  ejbPassivate
      ReturnValue       :
      TranStatus        :STATUS_ACTIVE
10/12/2001 13:59:19.940 : Call   :JcccCmpCBMRW  afterCompletion
      Param             :
      TranStatus        :STATUS_COMMITTING
10/12/2001 13:59:19.940 : Return   :JcccCmpCBMRW  afterCompletion
      ReturnValue       :
      TranStatus        :STATUS_COMMITTING
10/12/2001 13:59:19.940 : Return   :javax.transaction.UserTransaction  commit
      ReturnValue       :
      TranStatus        :STATUS_NO_TRANSACTION
10/12/2001 13:59:19.940 : Return   :JcccCmpCBSRW  empUpdate
      ReturnValue       :(int)1
      TranStatus        :STATUS_NO_TRANSACTION
10/12/2001 13:59:26.379 : Call   :JcccCmpEntityUserExRW  unsetEntityContext
      Param             :
      TranStatus        :STATUS_NO_TRANSACTION
10/12/2001 13:59:26.379 : Return   :JcccCmpEntityUserExRW  unsetEntityContext
      ReturnValue       :
      TranStatus        :STATUS_NO_TRANSACTION
10/12/2001 13:59:26.379 : Call   :JcccCmpEntityUserExRW  unsetEntityContext
      Param             :
      TranStatus        :STATUS_NO_TRANSACTION
10/12/2001 13:59:26.379 : Return   :JcccCmpEntityUserExRW  unsetEntityContext
      ReturnValue       :
      TranStatus        :STATUS_NO_TRANSACTION
10/12/2001 13:59:26.379 : Call   :JcccCmpEntityUserExRW  unsetEntityContext
      Param             :
      TranStatus        :STATUS_NO_TRANSACTION
10/12/2001 13:59:26.379 : Return   :JcccCmpEntityUserExRW  unsetEntityContext
      ReturnValue       :
      TranStatus        :STATUS_NO_TRANSACTION
10/12/2001 13:59:26.379 : Call   :JcccCmpEntityUserExRW  unsetEntityContext
      Param             :
      TranStatus        :STATUS_NO_TRANSACTION
10/12/2001 13:59:26.379 : Return   :JcccCmpEntityUserExRW  unsetEntityContext
      ReturnValue       :
      TranStatus        :STATUS_NO_TRANSACTION
10/12/2001 13:59:26.379 : Call   :JcccCmpEntityUserExRW  unsetEntityContext
      Param             :
      TranStatus        :STATUS_NO_TRANSACTION
10/12/2001 13:59:26.379 : Return   :JcccCmpEntityUserExRW  unsetEntityContext
      ReturnValue       :
      TranStatus        :STATUS_NO_TRANSACTION
10/12/2001 13:59:26.389 : Call   :JcccCmpEntityUserExRW  unsetEntityContext
```

```
      Param              :
      TranStatus         :STATUS_NO_TRANSACTION
10/12/2001 13:59:26.389 : Return   :JcccCmpEntityUserExRW  unsetEntityContext
      ReturnValue        :
      TranStatus         :STATUS_NO_TRANSACTION
10/12/2001 13:59:26.389 : Call   :JcccCmpEntityUserExRW  unsetEntityContext
      Param              :
      TranStatus         :STATUS_NO_TRANSACTION
10/12/2001 13:59:26.389 : Return   :JcccCmpEntityUserExRW  unsetEntityContext
      ReturnValue        :
      TranStatus         :STATUS_NO_TRANSACTION
10/12/2001 13:59:26.389 : Call   :JcccCmpEntityUserExRW  unsetEntityContext
      Param              :
      TranStatus         :STATUS_NO_TRANSACTION
10/12/2001 13:59:26.389 : Return   :JcccCmpEntityUserExRW  unsetEntityContext
      ReturnValue        :
      TranStatus         :STATUS_NO_TRANSACTION
10/12/2001 13:59:26.389 : Call   :JcccCmpEntityUserExRW  unsetEntityContext
      Param              :
      TranStatus         :STATUS_NO_TRANSACTION
10/12/2001 13:59:26.389 : Return   :JcccCmpEntityUserExRW  unsetEntityContext
      ReturnValue        :
      TranStatus         :STATUS_NO_TRANSACTION
10/12/2001 13:59:26.389 : Call   :JcccCmpEntityUserExRW  unsetEntityContext
      Param              :
      TranStatus         :STATUS_NO_TRANSACTION
10/12/2001 13:59:26.389 : Return   :JcccCmpEntityUserExRW  unsetEntityContext
      ReturnValue        :
      TranStatus         :STATUS_NO_TRANSACTION
10/12/2001 13:59:26.389 : Call   :JcccCmpEntityRW  unsetEntityContext
      Param              :
      TranStatus         :STATUS_NO_TRANSACTION
10/12/2001 13:59:26.389 : Return   :JcccCmpEntityRW  unsetEntityContext
      ReturnValue        :
      TranStatus         :STATUS_NO_TRANSACTION
10/12/2001 13:59:26.389 : Call   :JcccCmpEntityRW  unsetEntityContext
      Param              :
      TranStatus         :STATUS_NO_TRANSACTION
10/12/2001 13:59:26.389 : Return   :JcccCmpEntityRW  unsetEntityContext
      ReturnValue        :
      TranStatus         :STATUS_NO_TRANSACTION
10/12/2001 13:59:26.389 : Call   :JcccCmpEntityRW  unsetEntityContext
      Param              :
      TranStatus         :STATUS_NO_TRANSACTION
10/12/2001 13:59:26.389 : Return   :JcccCmpEntityRW  unsetEntityContext
      ReturnValue        :
      TranStatus         :STATUS_NO_TRANSACTION
10/12/2001 13:59:26.389 : Call   :JcccCmpEntityRW  unsetEntityContext
      Param              :
      TranStatus         :STATUS_NO_TRANSACTION
10/12/2001 13:59:26.389 : Return   :JcccCmpEntityRW  unsetEntityContext
      ReturnValue        :
      TranStatus         :STATUS_NO_TRANSACTION
10/12/2001 13:59:26.389 : Call   :JcccCmpEntityRW  unsetEntityContext
```

```
      Param             :
      TranStatus        :STATUS_NO_TRANSACTION
10/12/2001 13:59:26.389 : Return   :JcccCmpEntityRW  unsetEntityContext
      ReturnValue       :
      TranStatus        :STATUS_NO_TRANSACTION
10/12/2001 13:59:26.389 : Call   :JcccCmpEntityRW  unsetEntityContext
      Param             :
      TranStatus        :STATUS_NO_TRANSACTION
10/12/2001 13:59:26.389 : Return   :JcccCmpEntityRW  unsetEntityContext
      ReturnValue       :
      TranStatus        :STATUS_NO_TRANSACTION
10/12/2001 13:59:26.389 : Call   :JcccCmpEntityRW  unsetEntityContext
      Param             :
      TranStatus        :STATUS_NO_TRANSACTION
10/12/2001 13:59:26.399 : Return   :JcccCmpEntityRW  unsetEntityContext
      ReturnValue       :
      TranStatus        :STATUS_NO_TRANSACTION
10/12/2001 13:59:26.399 : Call   :JcccCmpEntityRW  unsetEntityContext
      Param             :
      TranStatus        :STATUS_NO_TRANSACTION
10/12/2001 13:59:26.399 : Return   :JcccCmpEntityRW  unsetEntityContext
      ReturnValue       :
      TranStatus        :STATUS_NO_TRANSACTION
10/12/2001 13:59:26.399 : Call   :JcccCmpEntityRW  unsetEntityContext
      Param             :
      TranStatus        :STATUS_NO_TRANSACTION
10/12/2001 13:59:26.399 : Return   :JcccCmpEntityRW  unsetEntityContext
      ReturnValue       :
      TranStatus        :STATUS_NO_TRANSACTION
10/12/2001 13:59:26.399 : Call   :JcccCmpEntityRW  unsetEntityContext
      Param             :
      TranStatus        :STATUS_NO_TRANSACTION
10/12/2001 13:59:26.399 : Return   :JcccCmpEntityRW  unsetEntityContext
      ReturnValue       :
      TranStatus        :STATUS_NO_TRANSACTION
10/12/2001 13:59:26.399 : Call   :JcccCmpEntitySysExRW  unsetEntityContext
      Param             :
      TranStatus        :STATUS_NO_TRANSACTION
10/12/2001 13:59:26.399 : Return   :JcccCmpEntitySysExRW  unsetEntityContext
      ReturnValue       :
      TranStatus        :STATUS_NO_TRANSACTION
10/12/2001 13:59:26.399 : Call   :JcccCmpEntitySysExRW  unsetEntityContext
      Param             :
      TranStatus        :STATUS_NO_TRANSACTION
10/12/2001 13:59:26.399 : Return   :JcccCmpEntitySysExRW  unsetEntityContext
      ReturnValue       :
      TranStatus        :STATUS_NO_TRANSACTION
10/12/2001 13:59:26.409 : Call   :JcccCmpEntitySysExRW  unsetEntityContext
      Param             :
      TranStatus        :STATUS_NO_TRANSACTION
10/12/2001 13:59:26.409 : Return   :JcccCmpEntitySysExRW  unsetEntityContext
      ReturnValue       :
      TranStatus        :STATUS_NO_TRANSACTION
10/12/2001 13:59:26.409 : Call   :JcccCmpEntitySysExRW  unsetEntityContext
```

```
      Param             :
      TranStatus        :STATUS_NO_TRANSACTION
10/12/2001 13:59:26.409 : Return   :JcccCmpEntitySysExRW  unsetEntityContext
      ReturnValue       :
      TranStatus        :STATUS_NO_TRANSACTION
10/12/2001 13:59:26.409 : Call   :JcccCmpEntitySysExRW  unsetEntityContext
      Param             :
      TranStatus        :STATUS_NO_TRANSACTION
10/12/2001 13:59:26.409 : Return   :JcccCmpEntitySysExRW  unsetEntityContext
      ReturnValue       :
      TranStatus        :STATUS_NO_TRANSACTION
10/12/2001 13:59:26.409 : Call   :JcccCmpEntitySysExRW  unsetEntityContext
      Param             :
      TranStatus        :STATUS_NO_TRANSACTION
10/12/2001 13:59:26.409 : Return   :JcccCmpEntitySysExRW  unsetEntityContext
      ReturnValue       :
      TranStatus        :STATUS_NO_TRANSACTION
10/12/2001 13:59:26.409 : Call   :JcccCmpEntitySysExRW  unsetEntityContext
      Param             :
      TranStatus        :STATUS_NO_TRANSACTION
10/12/2001 13:59:26.409 : Return   :JcccCmpEntitySysExRW  unsetEntityContext
      ReturnValue       :
      TranStatus        :STATUS_NO_TRANSACTION
10/12/2001 13:59:26.409 : Call   :JcccCmpEntitySysExRW  unsetEntityContext
      Param             :
      TranStatus        :STATUS_NO_TRANSACTION
10/12/2001 13:59:26.409 : Return   :JcccCmpEntitySysExRW  unsetEntityContext
      ReturnValue       :
      TranStatus        :STATUS_NO_TRANSACTION
10/12/2001 13:59:26.409 : Call   :JcccCmpEntitySysExRW  unsetEntityContext
      Param             :
      TranStatus        :STATUS_NO_TRANSACTION
10/12/2001 13:59:26.409 : Return   :JcccCmpEntitySysExRW  unsetEntityContext
      ReturnValue       :
      TranStatus        :STATUS_NO_TRANSACTION
10/12/2001 13:59:26.409 : Call   :JcccCmpEntitySysExRW  unsetEntityContext
      Param             :
      TranStatus        :STATUS_NO_TRANSACTION
10/12/2001 13:59:26.409 : Return   :JcccCmpEntitySysExRW  unsetEntityContext
      ReturnValue       :
      TranStatus        :STATUS_NO_TRANSACTION
```

Using Application Debugging Information

In this method, processes to output debugging information are defined beforehand and then EJB applications are debugged according to that information during development.

Debugging Information

Standard output or standard error output is used as debugging information from the application.

Debug information is output to the IJServer log file.

Refer to 'IJServer log' for information on the IJServer log.

Output of Exception Information to the Standard Error Output

By using the printStackTrace() method of the exception class, the location where an error occurred can be determined to some degree.

Example

```
catch(Exception e)
{
String emsg = e.getMessage();
Systemerr.println("SampleBean.ejbCreate:Exception occurred;");
e.printStackTrace();
throw new javax.ejb.EJBException(emsg);
}
```

Note

- Standard output and standard error output will not take place if there is insufficient disk space on the destination disk, so ensure there is enough space available. In addition, use Microsoft Windows Explorer or an alternative file management tool to delete standard output and standard error output files that are no longer required.

Using the Debugger

This method employs the debugger provided by Apworks.

The debugger allows logical errors in the processing of a developed application to be detected while the application is being run.

Debugging is normally carried out by setting a breakpoint within the program source code and then referencing or changing variables while the program is stopped at the breakpoint.

For details on how to use the debugger, refer to the Apworks Apdesigner Programmer's Guide or the Component Designer User's Guide (note that this manual is not provided for the Plus Developer).

Note

When IJServer is debugged, two or more processes cannot run concurrently. Be sure to set 1 for 'Process concurrency' of the WorkUnit.

Automatic Thread Dump Collection

A thread dump is automatically collected when an application has caused a timeout or returns no response. The cause of no response of the application or process bottlenecks can be detected by checking the collected thread dump.

A thread dump is output to the container information log (info.log).

A thread dump is collected twice at 10-second intervals on the occurrence of one of the following events at which a thread dump is collected. The thread dumps collected indicate there is a problem in an application running on the thread that showed no change between the two dumps.

No thread dump is collected for 10 minutes after a thread dump is collected.

Event at which a thread dump is collected

- Timeout during startup

  There may be a problem in start-time execution class or init processing, or too much time may be taken for processing. The timeout value can be set at 'WorkUnit start wait time'.

- Application timeout

  The application may have a problem or take too much time for processing. The timeout value can be set at 'Maximum processing time of application'.

- Timeout during termination

  A thread dump is automatically collected if 'Process forced stop time' is exceeded while the IJServer WorkUnit is forcibly terminated.

  A thread dump is also collected if 'Process forced stop time' is exceeded during forced termination after IJServer WorkUnit termination stops responding. In this case, the stop-time execution class that works for stopping the IJServer WorkUnit may have a problem, the destroy processing may have a problem, or either processing takes too much time.

  The timeout value can be set at 'Process forced stop time'.

Debugging using Java Method Trace

The Java method trace function can be used for debugging.

This function collects method level trace of the J2EE application. This trace provides information useful for checking how far the application has processed normally or in which processing a termination or abnormality occurred.

Chapter 3

JNDI

This chapter explains how to use JNDI.

Environment set up when JNDI service provider is used

In Interstage, a JNDI service provider to enable access to J2EE objects is implemented. Refer to "JNDI Service Provider Environment Setup" for details of the environment set up when using the JNDI service provider implemented by Interstage.

Environment setup to enable various objects to be referenced

When J2EE's various resources and EJB applications are accessed from the J2EE application, a JNDI interface is used. The objects that can be referenced in JNDI are as follows.

Category	Object name	Web application	EJB application	J2EE application client	Applet
EJB object	EJB Home object	O	O	O	O
	EJB Local Home object	O	O	X	X
Resource reference	JDBC data source	O	O	O	X
	JMS connection factory	O	O	O	X
	JavaMail mail session	O	O	O	X
	URL (Uniform Resource Locator)	O	O	O	X
	connector connection factory	X	O	X	X
Resource environment reference	JMS Destination	O	O	O	X

Category	Object name	Web application	EJB application	J2EE application client	Applet
others	Environment entry	O	O	O	X
	UserTransaction	O	O	O	X
	ORB	O	O	O	O

Writing to "deployment descriptor"

Write the information of objects to be referenced to the deployment descriptor file of each application. Refer to "Description in deployment descriptor file" for details of writing to the deployment descriptor file.

In the case of Applet, only EJB can be referenced; therefore a deployment descriptor does not need to be specified. Only by specifying an EJB application name in the lookup argument, EJB Home objects can be referenced.

Referencing objects

An object contained in the deployment descriptor can be referenced from a J2EE application using the JNDI interface's lookup method. Refer to "Referencing Objects" for details.

Name conversion function

If an application's description differs from the operation environment's real name, a deployment descriptor's reference name and the operation environment's real name can be correlated by using the name conversion function. In this way, if the relationship between the deployment descriptor's reference name and operation environment's real name is defined by using the name conversion file, an application independent of the operation environment can be created. Refer to "Name Conversion Function" for details of the name conversion function.

Setting when Fujitsu XML processor is used

For usual analysis of the deployment descriptor file or name conversion file, use Sun's XML parser. When Fujitsu XML processor is used for the analysis processing, refer to "Setting for using the Fujitsu XML processor" in Customizing and Checking the Operating Environment in Chapter 2.

Using UserTransaction

The transaction start and end can be controlled by using a looked up UserTransaction object. Refer to "Transaction Function using the UserTransaction Interface" for details of using UserTransaction.

JNDI Service Provider Environment Setup

This section explains the environment set up when the JNDI service provider provided by Interstage (JNDI SP) is used.

- **Web application**

  Because the JNDI SP operates by default, an environment setup is not required. However, it is required when HTTP tunneling is used. Refer to Web application.

- **EJB application**

  Since JNDI SP of Interstage operates by the default, there is no necessity for an environmental setup.

- **J2EE application client**

  For details on the setting required for J2EE application clients, refer to "J2EE application client".

- **Applet**

  Refer to "Applet" for details of the setting required for Applet.

Web application

When the HTTP tunneling is used in the Web application using JNDI, set JNDI environment properties. Specify these properties as explained below.

- FJjndi.properties file

  Deploy the FJjndi.properties file in the following.

  C:\Interstage\J2EE\etc

  /etc/opt/FJSVj2ee/etc

  /etc/opt/FJSVj2ee/etc

- Setting by IJServer JavaVM option

  Set the properties in the IJServer JavaVM option. Make this setting on the Interstage Management Console.

When an environment property is specified more than once, "IJServer JavaVM option settings" are enabled.

The values that can be set in an environment property are given below. The character strings of the environment property are case-sensitive.

Environment property	Value	Setting contents	OS applied
HTTPGW	Refer to "J2EE application client" environment property "HTTPGW".		

J2EE application client

For a J2EE application client, set a JNDI environment property.

The JNDI environment property is created when new javax.naming.InitialContext() is used to access the JNDI SP from the application. It is an environment property used in the InitialContext initialization processing. The environment property is specified in the following files or arguments.

- jndi.properties file

  Only the environment property "java.naming.factory.initial" can be specified.

  The jndi.properties file must be located in the archive specified by the class path or in lib under the directory set in a Java system property "java.home".

  When JBK is installed, locate the jndi.properties file as explained below.

  Windows

  <JBK installation directory>\jdk\jre\lib

  Solaris OE Linux

  <JBK installation directory>/jdk13/jre/lib

- FJjndi.properties file

  Environment properties "java.naming.factory.initial" and "com.fujitsu.interstage.isas.SystemName" cannot be specified.

  Deploy the FJjndi.properties file in the following.

  Windows

  C:\Interstage\J2EE\etc

  Solaris OE

  /etc/opt/FJSVj2ee/etc

  When a system name (environment property: com.fujitsu.interstage.isas.SystemName) is specified in one of the following, deploy the FJjndi.properties file in the following directory:

  /var/opt/FJSVisas/system/system name/FJSVj2ee/etc/

  – The argument environment in new javax.naming.InitialContext(Hashtable environment)
  – Command line argument (-D) when the application is started

  Linux

  /etc/opt/FJSVj2ee/etc

- The argument in javax.naming.InitialContext(Hashtable environment)
- Command line argument (-D) when the application is started

When an environment property is specified more than once, they are overwritten in the following order (the environment property specified by 3 is given the top priority).

1. FJjndi.properties file
2. The argument in javax.naming.InitialContext(Hashtable environment)
3. Command line argument (-D) when the application is started

The environment property "java.naming.factory.initial" is overwritten in the following order (the environment property specified by 3 is given the top priority).

1. jndi.properties file
2. Command line argument (-D) when the application is started
3. The argument in javax.naming.InitialContext(Hashtable environment)

The values that can be set in an environment property are given in the following table.

Windows

The character strings of the environment property and "java.naming.factory.initial" values are case-sensitive.

Solaris OE Linux

The character strings of the environment property and values are case-sensitive.

The "VerificationMethod" and "com.fujitsu.ObjectDirector.CORBA.GlobalTransactionMode" values are excluded.

Environment property	Value	Setting contents	OS applied
java.naming.factory.initial (*1)	com.fujitsu.interstage.j2ee.jndi.InitialContextFactoryForClient	The InitialContext factory class name to access JNDI SP is specified.	All
FJUserID (*2)	Optional character string	A user name used in Smart Repository user authentication is specified. If this value is omitted, the user authentication is not performed.	All
FJPassword (*2)	Optional character string	A Password used in Smart Repository user authentication is specified. If this value is omitted, the user authentication is not performed.	All
com.fujitsu.interstage.j2ee.DeploymentDescriptorClient (*3)	Optional character string	The J2EE application client deployment descriptor file name is specified in full path.	All
EBEproperties	Optional character string	A name conversion file name is specified (full path specification is disabled). If this value is omitted, the name conversion is not performed.	All

Environment property	Value	Setting contents	OS applied
HTTPGW	(1) For Interstage HTTP Server [Format for using SSL communication] https://host-name/url-name [Format for not using SSL communication] http://host-name/url-name host-name: Specifies the Web server that downloads HTML. url-name: Specifies od-httpgw. For url-name, specify the URL of the Location directive. Windows Solaris OE (2) For non-Interstage HTTP server [Format for using SSL communication] https://host-name/cgi-ID/gateway-name [Format for not using SSL communication] http://host-name/cgi-ID/gateway-name host-name: Specifies the Web server that downloads HTML. cgi-ID: When the InfoProvider Pro is used, specify the cgi identification name. Windows When Internet Information Server is used, specify "Virtual directory" alias name. gateway-name: Windows	The gateway that processes the HTTP tunneling is specified. If this value is omitted, the HTTP tunneling is not used. (*5)	All (*4)

Environment property	Value	Setting contents	OS applied
	Specifies ODhttp.dll (HTTP gateway). Solaris OE Specifies libOMhttp.so (HTTP-IIOP gateway).		
VerificationMethod	- Well-formed Verifies only whether the XML document is well-formed or not. - DTD Verifies whether the XML document is well-formed or not. Verification by DTD is also performed. (Default)	Verification method by deployment descriptor and name conversion files' parser is specified.	All
com.fujitsu.ObjectDirector.CORBA.GlobalTransactionMode	- True Performed. - False Not performed	Specify whether distributed transaction control is performed or not.	All (*4)
com.fujitsu.interstage.isas.SystemName	Optional character string	When an extended system is created, the system name to be operated is specified. If this value is omitted, the default system is operated.	Solaris OE (*4)

*1 This environment property must be specified in the jndi.properties file, new javax.naming.InitialContext(Hashtable environment) argument environment, or command line's argument (-D) when the application is started.

*2 FJUserID and FJPassword must be specified together.

*3 This environment property must be specified in the FJjndi.properties file, new javax.naming.InitialContext(Hashtable environment) argument environment, or command line's argument (-D) when the application is started.

*4 Web-J Edition is not supported.

*5 The format of a host name that can be specified as an argument to be passed to "-ORB_FJ_HTTPGW" is shown below.

(1) For Interstage HTTP Server

```
http://ipv4address_host-name/url-name
http://ipv4address_host-name:Port_number/url-name
https://ipv4address_host-name/url-name
https://ipv4address_host-name:Port_number/url-name
```

(2) For other than Interstage HTTP Server

```
http://IPv4-address-host-name/cgi-identification-name/gateway-name
http://IPv4-address-host-name:Port_number/cgi-identification-name/gateway
-name
http://[IPv6-address]/cgi-identification-name/gateway-name
http://[IPv6-address]:Port_number/cgi-identification-name/gateway-name
https://IPv4-address-host-name/cgi-identification-name/gateway-name
https://IPv4-address-host-name:Port_number/cgi-identification-name/
gateway-name
```

When using an address in the IPv6 format, it needs to be enclosed by square brackets ("[" and "]").

In addition, in an IPv6 environment, since the SSL function cannot be used, 'https' cannot be specified.

[Description example of jndi.properties file]

```
java.naming.factory.initial=com.fujitsu.interstage.j2ee.jndi.InitialConte
xtFactoryForClient
```

Note

The Java system property "java.home" can be changed by the command line argument indicated when the application is started. However, be very careful about the change because orb.properties may be unable to be referenced, causing an operation error.

Example Windows

```
java -Djava.home=x:\aaaa\bbbb myapplication
```

Example Solaris OE Linux

```
java -Djava.home=/aaaa/bbbb myapplication
```

[Description example of FJjndi.properties file]

Windows

```
deployment descriptor file name:C:\env\application-client.xml
Name Conversion file name      :ClientebeProperties.xml

com.fujitsu.interstage.j2ee.DeploymentDescriptorClient=C:\env\application
-client.xml
EBEproperties=ClientebeProperties.xml
```

Solaris OE

```
deployment descriptor file
name:/export/home/j2eeapl/application-client.xml
Name Conversion file name      :ClientebeProperties.xml

com.fujitsu.interstage.j2ee.DeploymentDescriptorClient=/export/home/j2eea
pl/application-client.xml
EBEproperties=ClientebeProperties.xml
```

Linux

```
deployment descriptor file name:/home/j2eeapl/application-client.xml
Name Conversion file name      :ClientebeProperties.xml

com.fujitsu.interstage.j2ee.DeploymentDescriptorClient=/home/j2eeapl/appl
ication-client.xml
EBEproperties=ClientebeProperties.xml
```

Applet

When a client application is developed as a Java Applet using an EJB client, it differs from a Java application in the following aspect.

- lookup processing to send inquires about the EJB application object location to the Naming Service

A description example of the lookup processing in Java Applet is given below.

```
  // Acquisition of InitialContext
  Hashtable env = new Hashtable();                                    ....1
  env.put("java.naming.factory.initial",
          "com.fujitsu.interstage.ejb.jndi.FJCNCtxFactoryForClient");   ....1
  env.put("java.naming.applet", this);                                ....1
  javax.naming.Context ic = new javax.naming.InitialContext( env);      ....2
  // lookup
  java.lang.Object Obj = (java.lang.Object)ic.lookup("SampleBean");     ....3
  // home's narrow()
  h = (SampleHome)javax.rmi.PortableRemoteObject.narrow( Obj,
SampleHome.class); ....4
```

1. Set environment information to make a context. Specify it as explained above.
2. Create a context for lookup. Specify it as explained above.
3. To make a lookup, specify the EJB application name in the argument. If the lookup fails, a javax.naming.NameNotFoundException exception occurs.
   The failure cause is indicated as the detail message of the exception.
4. To narrow down the looked up objects, issue a javax.rmi.PortableRemoteObject.narrow.

Notes

- Use the constructor to make the InitialContext acquisition (steps 1 to 2 above) only once in the application.
- When an Applet is used without a Portable-ORB, it is not possible to download the client distribution data of the EJB application to be used from the Web server and use it.

  Copy the client distribution data to the client environment and set the copy destination jar file or folder in CLASSPATH and use it.

Refer to "Using Java Applets" in Chapter 11 for details of developing Java Applets.

When the Portable-ORB is used, refer to "Environment setup in client environment".

When the Java Applet is executed, it is necessary to set the data in the policy file that the JBK plug-in uses for each execution machine. The data to be set is as follows.

- EJB service class
- JDK class

A setting example is given below. Refer to the “Apworks J Business Kit Online Manual” for details of the policy file that the JBK plug-in uses.

- EJB service class

```
(When Interstage install folder C:\Interstage is specified)
grant codeBase "file:/C:/Interstage/EJBCL/LIB/-" {
permission java.security.AllPermission;
};
```

- JDK class

```
(When Interstage install folder C:\Interstage is specified)

In the case of JDK1.3
grant codeBase "file:/C:/Interstage/JDK13/jre/lib/-" {
permission java.security.AllPermission;
};
In the case of JRE1.3
grant codeBase "file:/C:/Interstage/JRE13/lib/-" {
permission java.security.AllPermission;
};
In the case of JDK1.4
grant codeBase "file:/C:/Interstage/JDK14/jre/lib/-" {
permission java.security.AllPermission;
};
In the case of JRE1.4
grant codeBase "file:/C:/Interstage/JRE14/lib/-" {
permission java.security.AllPermission;
};
```

Note

Even when a Java application is executed in the same way as for earlier versions, EJB can be referenced by setting the following in the system properties. However, specify the JNDI SP as explained in the J2EE application client.

```
java.naming.factory.initial=com.fujitsu.interstage.ejb.jndi.FJCNCtxFactor
yForClient
```

Environment Setup for Referencing EJB

The EJB Local Home or EJB Home object of the deployed EJB application can be acquired.

The EJB Local Home object reference directly acquires the same IJServer's EJB application object.

The EJB Home object reference is acquired from the Naming Service.

Refer to "Deploying and Setting J2EE Applications" in Chapter 2 for details.

EJB to be deployed on the IJServer (Web + EJB[1VM]) is not added to the Naming Service. Therefore, EJB can be referenced only from the IJServer.

The following products acquire an object reference from the Naming Service that operates by default on the local machine if they are used on a server. To change the Naming Service to be referenced, select [System] > [Environment Settings] tab > [Detailed Setup] > [Detailed Settings] on the Interstage Management Console.

- Interstage Application Server Enterprise Edition
- Interstage Application Server Standard Edition
- Interstage Application Server Plus

For Interstage Application Server Web-J Edition and clients, the Naming Service to be referenced must be set. Refer to "inithost/initial_hosts" in Appendix A, “CORBA Service Environment Definition” in the “Tuning Guide” for details.

When Interstage client functions are installed, an environment setup is required. Refer to "Environment setup in client environment" for details. Note that the functions are automatically set when the IJServer is started up; therefore the environment setup is not required.

To acquire an EJB Home object reference in the following cases, the class path must be set to the client distribution data to be output when the EJB application to be referenced is deployed.

- When the EJB application deployed on another IJServer is looked up from the J2EE application deployed on the IJServer.
- When the EJB application is looked up from a J2EE application client or Applet

Setting client distribution data

The client distribution data output destination for each IJServer is as follows.

Set the EJB calling application class path.

When the EJB application deployment destination server machine differs from the EJB application calling machine, copy the client distribution data to the calling machine and set the class path to the copy destination file.

Windows

```
C:\Interstage\J2EE\var\deployment\ijserver\[IJServer
 name]\distribute\[jar file name]
```

Solaris OE Linux

```
/opt/FJSVj2ee/var/deployment/ijserver/[IJServer name]/distribute/[jar file name]
```

* [jar file name] is given by replacing the deployed ejb-jar file name "." with "_" and adding "_client.jar" to that name.

When the deployed ejb-jar file name is "Sample.jar," the client distribution jar name is to be "Sample_jar_client.jar".

Environment setup in client environment

When a J2EE application or Applet is operated, the following environment setup is required. Check the environment variable settings and set them as required.

Because the settings are automatically made at startup, the environment setup is not required for IJServer.

- CORBA service setting
- ORB specification
- Setting environment variables

CORBA service setting

After the CORBA service is installed, make the following information setting.

Host name definition

In the inithost file, define the Naming Service activated host name.

In the following file, the Naming Service located host name and port number must be included.

```
C:\Interstage\ODWIN\etc\inithost
```

[Definition method]

format:

host-name port-number

sample:

hostname 8002

"config" file modification

When the following statement in the config file is operated with the initial value added, add the following value.

Refer to the “Tuning Guide” for details of the initial value of the setting value.

Statement	Value of add
max_processes	Number of processes of client application to be added

ORB specification

As the environment setup to start an application, the ORB (Object Request Broker) to be used must be specified. Specify the ORB in one of the following methods:

- Specifying ORB when application is active
- Specifying ORB in environment setup file

Specify the ORB by setting the following values in the above method.

Property name	value
org.omg.CORBA.ORBClass	com.fujitsu.ObjectDirector.CORBA.ORB
org.omg.CORBA.ORBSingletonClass	com.fujitsu.ObjectDirector.CORBA.SingletonORB
javax.rmi.CORBA.StubClass	com.fujitsu.ObjectDirector.rmi.CORBA.StubDelegateImpl
javax.rmi.CORBA.UtilClass	com.fujitsu.ObjectDirector.rmi.CORBA.UtilDelegateImpl
javax.rmi.CORBA.PortableRemoteObjectClass	com.fujitsu.ObjectDirector.rmi.CORBA.PortableRemoteObjectDelegateImpl

Specifying ORB when application is active

When a Java application is executed, specify the ORB to be used as the java command parameter. Write the required information after the -D option as follows.

```
java -Dorg.omg.CORBA.ORBClass=com.fujitsu.ObjectDirector.CORBA.ORB
-Dorg.omg.CORBA.ORBSingletonClass=com.fujitsu.ObjectDirector.CORBA.Single
tonORB
-Djavax.rmi.CORBA.StubClass=com.fujitsu.ObjectDirector.rmi.CORBA.StubDele
gateImpl
-Djavax.rmi.CORBA.UtilClass=com.fujitsu.ObjectDirector.rmi.CORBA.UtilDele
gateImpl
-Djavax.rmi.CORBA.PortableRemoteObjectClass=com.fujitsu.ObjectDirector.rm
i.CORBA.PortableRemoteObjectDelegateImpl <application class name>
```

Specifying ORB in environment setup file

Create the text file containing the ORB to be used (file name: orb.properties) and store it in lib under the directory set in the Java system property "java.home".

When JBK is installed, the text file is stored in the following.

Interstage Apworks client operation package

<JBK-install-directory>\jre\lib (*1)

*1 For J2EE application client only

[Description example of orb.properties file]

```
org.omg.CORBA.ORBClass=com.fujitsu.ObjectDirector.CORBA.ORB
org.omg.CORBA.ORBSingletonClass=com.fujitsu.ObjectDirector.CORBA.SingletonORB
javax.rmi.CORBA.StubClass=com.fujitsu.ObjectDirector.rmi.CORBA.StubDelegateImpl
javax.rmi.CORBA.UtilClass=com.fujitsu.ObjectDirector.rmi.CORBA.UtilDelegateImpl
javax.rmi.CORBA.PortableRemoteObjectClass=com.fujitsu.ObjectDirector.rmi.CORBA.PortableRemoteObjectDelegateImpl
```

Setting environment variables

Windows

Environment variable	Setting value
CLASSPATH	C:\Interstage\J2EE\lib\isj2ee.jar (*1) C:\Interstage\ODWin\etc\class\ODjava2.jar (*2) C:\Interstage\ODWin\etc\class\ODjava4.jar (*3) C:\Interstage\EJB\lib\fjcontainer72.jar

Solaris OE

Environment variable	Setting value
CLASSPATH	/opt/FJSVisj2ee/lib/isj2ee.jar (*1) /opt/FSUNod/etc/class/ODjava2.jar (*2) /opt/FSUNod/etc/class/ODjava4.jar (*3) /opt/FJSVejb/lib/fjcontainer72.jar
LD_LIBRARY_PATH	/opt/FSUNod/lib

Linux

Environment variable	Setting value
CLASSPATH	/opt/FJSVisj2ee/lib/isj2ee.jar (*1) /opt/FJSVod/etc/class/ODjava2.jar (*2) /opt/FJSVod/etc/class/ODjava4.jar (*3) /opt/FJSVejb/lib/fjcontainer72.jar
LD_LIBRARY_PATH	/opt/FJSVod/lib

*1 Required for J2EE application client only

*2 In the case of JDK/JRE1.3

*3 In the case of JDK/JRE1.4

Environment Setup when JDBC (Database) is Referenced

Make the environment set up to use the JDBC data sources.

Setup of the class path

When you use JDBC, set up the class path as follows.

IJServer	Setting item
IJServer	J2EE property class path
	WorkUnit definition class path
IJServer that does not separate a class loader	J2EE property class path
	WorkUnit definition class path
	CLASSPATH environment variable

- Environment set up when Symfoware is used

  Note Windows Solaris OE

  The following contains the settings used when Symfoware V4.0L10 or later (Windows) and Symfoware 3.2 or later (Solaris OE) are used. Refer to the Symfoware Client JDBC driver's online manual for details of the environment setup to use earlier Symfoware versions.

- Environment set up when Oracle is used
- Environment set up when SQL Server is used

Environment set up when Symfoware is used

The following environment set up is required when Symfoware is used.

- Setting environment variable
- Starting JDBC Naming Service
- Adding JDBC data sources
- Resource access definition

When Symfoware on a non-Interstage server system is accessed, the operation below is also required.

Setting environment variable

Set the following item.

When a WorkUnit is used, set the environment variable in the WorkUnit definition on the Interstage Management Console. It is effective even if it is set as a system environment variable. The system environment variable CLASSPATH is only enabled when "Do not separate" is set for separation of IJServer class loaders.

Windows

Setting item	Symfoware version	Setting value
PATH	V4.0L10 or later, earlier than V5.0L10	JDBC driver install directory\com\fujitsu\symfoware\jdbc
	V5.0L10 or later	JDBC driver install directory\fjjdbc\bin
CLASSPATH	V4.0L10 or later, earlier than V5.0L10	JDBC driver install directory JDBC driver install directory\com\fujitsu\symfoware\jdbc2
	V5.0L10 or later	JDBC driver install directory\fjjdbc\lib\fjsymjdbc2.jar

When the client function is being used

Sample: When Symfoware JDBC driver install directory is C:\classes\SymfoJDBC

In the case of Symfoware V4.0L10

```
set PATH=C:\classes\SymfoJDBC\com\fujitsu\symfoware\jdbc;%PATH%
set CLASSPATH=C:\classes\SymfoJDBC;%CLASSPATH%
set
CLASSPATH=C:\classes\SymfoJDBC\com\fujitsu\symfoware\jdbc2;%CLASSPATH%
```

In the case of Symfoware V5.0L10

```
set PATH=C:\classes\SymfoJDBC\fjjdbc\bin;%PATH%
set CLASSPATH=C:\classes\SymfoJDBC\fjjdbc\lib\fjsymjdbc2.jar;%CLASSPATH%
```

Solaris OE Linux

Setting item	Symfoware version	Setting value
LIBRARY PATH	3.2 or later, earlier than 5.0	Solaris OE Symfoware install directory/lib ICONV install directory/lib JDBC driver install directory/com/fujitsu/symfoware/jdbc
	5.0 or later	Solaris OE Symfoware install directory/lib ICONV install directory/lib JDBC driver install directory/fjjdbc/bin Linux Symfoware install directory/FJSVrdb2b/lib ICONV install directory/FSUNiconv/lib FJSVsymjd package install directory/FJSVsymjd/fjjdbc/bin

Setting item	Symfoware version	Setting value
CLASSPATH	3.2 or later, earlier than 5.0	Solaris OE JDBC driver install directory JDBC driver install directory/com/fujitsu/symfoware/jdbc2
	5.0 or later	Solaris OE JDBC driver install directory/fjjdbc/lib/fjsymjdbc2.jar Linux FJSVsymjd package install directory/FJSVsymjd/fjjdbc/lib/fjsymjdbc2.jar

When the client function is being used

Sample: Solaris OE When Symfoware JDBC driver install directory is /classes/SymfoJDBC

In the case of Symfoware 3.2 (When using the C shell)

```
setenv LD_LIBRARY_PATH /opt/FSUNrdb2b/lib:${LD_LIBRARY_PATH}
setenv LD_LIBRARY_PATH /opt/FSUNiconv/lib:${LD_LIBRARY_PATH}
setenv LD_LIBRARY_PATH
/classes/SymfoJDBC/com/fujitsu/symfoware/jdbc:${LD_LIBRARY_PATH}
setenv CLASSPATH /classes/SymfoJDBC:${CLASSPATH}
setenv CLASSPATH
/classes/SymfoJDBC/com/fujitsu/symfoware/jdbc2:${CLASSPATH}
```

In the case of Symfoware 5.0 (When using the C shell)

```
setenv LD_LIBRARY_PATH /opt/FSUNrdb2b/lib:${LD_LIBRARY_PATH}
setenv LD_LIBRARY_PATH /opt/FSUNiconv/lib:${LD_LIBRARY_PATH}
setenv LD_LIBRARY_PATH /classes/SymfoJDBC/fjjdbc/bin:${LD_LIBRARY_PATH}
setenv CLASSPATH /classes/SymfoJDBC/fjjdbc/lib/fjsymjdbc2.jar:${CLASSPATH}
```

Sample: Linux When FJSVsymjd package is installed in /opt

In the case of Symfoware 5.0 (When using the C shell)

```
setenv LD_LIBRARY_PATH /opt/FJSVrdb2b/lib:${LD_LIBRARY_PATH}
setenv LD_LIBRARY_PATH /etc/opt/FSUNiconv/lib:${LD_LIBRARY_PATH}
setenv LD_LIBRARY_PATH /opt/FJSVsymjd/fjjdbc/bin:${LD_LIBRARY_PATH}
setenv CLASSPATH /opt/FJSVsymjd/fjjdbc/lib/fjsymjdbc2.jar:${CLASSPATH}
```

Starting JDBC Naming Service

When the JDBC is used, the Naming Service must be started to add to and view the JDBC data sources as well as to execute J2EE applications.

Start the Naming Service in the following procedure.

1. Check Java environment

   Check whether the Java environment is set up. For details of the Java environment setup, refer to "Environment Setup of Java" in “Checking the operating environment” in Chapter 2.

2. Check environment variable

   Set the environment variable specified in "Setting environment variable".

3. Start Naming Service

   Start the Naming Service.

   Sample:

```
java com.fujitsu.symfoware.jdbc2.naming.SYMNameService
```

Note

When using name conversion, if JDBC naming service has not started, an error will occur and the following messages will be output. Check the JNDI name (%s).

```
javax.naming.InvalidNameException physical-name is invalid NAME = %s
```

For details, refer to "Management when an Exception occurs in Lookup Processing" in Chapter 39 of "Messages".

Adding JDBC data sources

When the JDBC is used, JDBC data sources must be added.

Add the JDBC data sources in the following procedure.

1. Check Java environment

   Check whether the Java environment is set up. For details of setting up the Java environment, refer to "Environment Setup of Java" in “Checking the operating environment” in Chapter 2.

2. Check environment variable

   Set the environment variable specified in "Setting environment variable".

3. Start JDBC data source registration tool

   Use the user ID with administrator permission to activate the JDBC data source registration tool.

   Sample:

```
java com.fujitsu.symfoware.jdbc2.tool.FJJdbcTool
```

4. Add JDBC data sources

   Click the data source list's [Add] button, then set the information required to add data sources, and click the [OK] button.

   Sample:

Data source name	DS1
Protocol	local
Data resource name	DB1
User name	j2ee
Password	j2ee

Resource access definition

Make the resource access definition on the Interstage Management Console.

When Interstage client functions are installed, use the J2EE resource access definition. Refer to Chapter 19, "J2EE Resource Access Definition" for details.

Symfoware installed server system's environment setup

The connection type used to access Symfoware on a non-Interstage server system is called "RDB2_TCP".

The following operations are required to connect to Symfoware in RDB_TCP but not to access the same server system as Interstage.

- RDB2_TCP connection parameter setting
- RDB2_TCP port number setting

RDB2_TCP connection parameter setting

Add the following RDB2_TCP connection parameter to the Symfoware system operation environment file.

MAX_CONNECT_TCP = (n)

n: Maximum number of connection (0 by default)

The system operation environment file is stored in the location specified when Symfoware is installed. If the storage location is not specified, the system operation environment file is stored in the following location.

Windows

```
Symfoware install drive:\SFWETC\RDB\ETC\UXPSQLENV
```

Solaris OE

```
/opt/FSUNrdb2b/etc/fssqlenv
```

Linux

```
/opt/FJSVrdb2b/etc/fssqlenv
```

Note

When the system operation environment file does not contain MAX_CONNECT_TCP or MAX_CONNECT_TCP is specified by 0, a Symfoware ODBC driver error occurs when a J2EE application is executed. Refer to "Exceptions Output during J2EE Usage" in the Messages for details of the output exception information.

Setting RDB2_TCP port number

Set the RDB2_TCP port number in the following file.

Windows

```
In the case of Windows(R) 2000
Windows install directory\WINNT\system32\drivers\etc\services
```

Solaris OE Linux

```
/etc/services
```

Sample : When 2,050 is allocated to port number

```
RDBII  2050/TCP
```

Environment set up when Oracle is used

The following environment set up is required when Oracle is used.

Note

If Oracle10g or later is used in a Solaris OE 32 bit environment, use a 32 bit compatible library for creating and executing programs. Additionally, set the library path of the environment variable so that the 32 bit compatible library is enabled.

- Setting environment variable
- Resource access definition

Note

If Oracle10g or later is used in a 32 bit edition environment, use a 32 bit compatible library for creating and executing the program. Additionally, when executing the program, set PATH (Windows(R)), or the LD_LIBRARY_PATH environment variable (Solaris OE, Linux) to enable the 32 bit compatible library.

Setting environment variable

Set the following item.

When a WorkUnit is used, set the environment variable in the WorkUnit definition on the Interstage Management Console. It is effective even if it is set as a system environment variable. The system environment variable CLASSPATH is only enabled when "Do not separate" is set for separation of IJServer class loaders.

Set the class path in the Oracle JDBC driver jar file.

Setting item	Oracle version	JDK/JRE	**Setting value (*1)**
CLASSPATH	Oracle9i or earlier	1.3	Oracle install directory\jdbc\lib\classes12.jar or Oracle install directory\jdbc\lib\classes12.zip
			Oracle install directory\jdbc\lib\nls_charset12.jar or Oracle install directory\jdbc\lib\nls_charset12.zip
		1.4	Oracle install directory\jdbc\lib\ojdbc14.jar
			Oracle install directory\jdbc\lib\nls_charset12.jar or Oracle install directory\jdbc\lib\nls_charset12.zip
	Oracle10g or later	1.3	Oracle install directory\jdbc\lib|classes12.jar or Oracle install directory\jdbc\lib\classes12.zip
			Oracle install directory\jdbc\lib\orai18n.jar
		1.4	Oracle install directory\jdbc\lib\ojdbc14.jar
			Oracle install directory\jdbc\lib\orai18n.jar

*1 For Solaris OE or Linux, replace "\" with "/".

Setting when OCI driver is used

When the OCI driver is used, the following setting is required in addition to the above setting.

Windows

Setting item	Oracle version	**Setting value (*1)**
PATH	Common	Oracle install directory\bin

Solaris OE Linux

Setting item	Oracle version	**Setting value (*1)**
LD_LIBRARY_PATH	Oracle9i or earlier	Oracle install directory\\lib
	Oracle10g or later	Oracle install directory\lib
		Oracle install directory\lib32 (*2)
ORACLE_HOME	Common	Oracle install directory

*1 For Solaris OE or Linux, replace "\" with "/".

*2 This is the value that is set if a 32 bit compatible library is used in a Solaris OE 64 bit edition.

Setting File System Service Provider

It is necessary to download the File System Service Provider from Sun Microsystems, Inc.'s website and set the environment variable, but not when the downloaded version provided by Interstage is used. When the user wants to use the latest jar file, use the downloaded version.

Setting item	Setting value (*1)
CLASSPATH	The storage directory of the downloaded class\providerutil.jar
	The storage directory of the downloaded class\fscontext.jar

*1 For Solaris OE or Linux, replace "\" with "/".

*2 When using an ext directory, refer to "IJServer file configuration" in Chapter 1.

The class file provided by Interstage is stored in the following.

Windows

```
C:\Interstage\J2EE\lib\providerutil.jar
C:\Interstage\J2EE\lib\fscontext.jar
```

Solaris OE Linux

```
/opt/FJSVj2ee/lib/providerutil.jar
/opt/FJSVj2ee/lib/fscontext.jar
```

[Example of setting environment variable in system environment variable in Windows] Windows

1. Select [Control Panel] > [System] > [Details] and click the environment variable button.

   Note:

   This explanation is for Windows(R) 2000. The operation method depends on the OS to be used. For Windows(R) 9x and Windows(R) Me, edit autoexec.bat.

2. Add the setting value indicated above to the environment variable.

[Example of setting environment variable in system environment variable by using command]

In this example, providerutil.jar and fscontext.jar are downloaded from the website of Sun Microsystems, Inc. when the client function is used.

Windows

```
set CLASSPATH=%CLASSPATH%;%ORACLE_HOME%\jdbc\lib\classes12.zip
set CLASSPATH=%CLASSPATH%;%ORACLE_HOME%\jdbc\lib\nls_charset12.zip
set CLASSPATH=C:\classes\providerutil.jar;%CLASSPATH%
set CLASSPATH=C:\classes\fscontext.jar;%CLASSPATH%
```

Solaris OE Linux

```
[When using the C shell]
setenv CLASSPATH ${CLASSPATH}:${ORACLE_HOME}/jdbc/lib/classes12.zip
setenv CLASSPATH ${CLASSPATH}:${ORACLE_HOME}/jdbc/lib/nls_charset12.zip
setenv CLASSPATH /classes/providerutil.jar:${CLASSPATH}
setenv CLASSPATH /classes/fscontext.jar:${CLASSPATH}
```

[Example of setting in J2EE property]

Select [System] > [Environment Settings] > [J2EE property] on the Interstage Management Console.

When Interstage client functions are installed, directly edit the isj2ee.properties file.

Windows

```
C:\Interstage\J2EE\etc\isj2ee.properties
```

Solaris OE Linux

```
/etc/opt/FJSVj2ee/etc/isj2ee.properties
```

Note:

When a JDBC object is referenced from EJB and Web applications, set the system environment variable .

Resource access definition

When Interstage client functions are installed, use the J2EE resource access definition Refer to Chapter 19, "J2EE Resource Access Definition" for details.

Environment set up when SQL Server is used

In this section, Microsoft(R) SQL Server(TM) 2000 Driver for Java (TM) Database Connectivity (JDBC) is referred to as Microsoft(R) JDBC driver.

Also, in this section, the environment setup for connecting with SQL Server using a Microsoft (R) JDBC driver is explained. When using an Interstage JDBC driver, refer to Methods of Connection to an SQL Server of Chapter14, Using the Interstage JDBC Driver.

When the SQL Server is used, set the environment in the following procedure.

1. Downloading and installing Microsoft(R) JDBC driver

   1) Download

      The Microsoft(R) JDBC driver is not included in Microsoft(R) SQL Server(TM) 2000.

   Download Microsoft (R) JDBC driver Service Pack2 or later from Microsoft Corporation's website.

2) Installation

Refer to the installation method described website on Microsoft Corporation’s website for details of the installation.

2. Data source definition by resource access definition

Use one of the following methods.

- Define the data source on the Interstage Management Console.
- Define the data source by using the fjj2eeadmin command. Refer to "fjj2eeadmin" in the “Reference Manual (Command Edition)” for details.

  When the Microsoft(R) JDBC driver is used to access the SQL Server from a J2EE application with Interstage client functions installed, use the fjj2eeadmin command.

Setting environment variable

The setup of the environment variable for using IJServer is explained below. Specify CLASSPATH as explained below.

When a WorkUnit is used, set the environment variable in the WorkUnit definition on the Interstage Management Console. It is effective even if it is set as a system environment variable. The system environment variable CLASSPATH is only enabled when "Do not separate" is set for separation of IJServer class loaders.

Setting item	Setting value (*1)
PATH	JDBC install directory\lib\msbase.jar
	JDBC install directory\lib\mssqlserver.jar
	JDBC install directory\lib\msutil.jar

*1 For Solaris OE or Linux, replace "\" with "/".

Setting File System Service Provider

It is necessary to download the File System Service Provider from Sun Microsystems, Inc.'s website and set the environment variable, but not when the downloaded version provided by Interstage is used. When the user wants to use the latest jar file, use the downloaded version.

Setting item	Setting value (*1)
CLASSPATH	The storage directory of the downloaded class\providerutil.jar
	The storage directory of the downloaded class\fscontext.jar

*1 For Solaris OE or Linux, replace "\" with "/".

*2 When using an ext directory, refer to "IJServer file configuration" in Chapter 1.

[Example of the J2EE settings properties]

Select [System] > [Environment Settings] > [J2EE property] on the Interstage Management Console.

When Interstage client functions are installed, directly edit the isj2ee.properties file.

Windows

```
C:\Interstage\J2EE\etc\isj2ee.properties
```

Solaris OE Linux

```
/etc/opt/FJSVj2ee/etc/isj2ee.properties
```

Resource access definition

When Interstage client functions are installed, use the J2EE resource access definition Refer to Chapter 19, "J2EE Resource Access Definition" for details.

Environment Setup when JMS is Referenced

When JMS is referenced, perform the following.

- Developing the J2EE Application
- Resource access definition

Refer to "Procedure for Using JMS" in Chapter 2 for details of the operation.

Developing the J2EE Application

Develop the following applications according to intended usage:

For J2EE Application Client

- Message listener application
- Applications that use the Durable Subscription function
- Applications that use the message nonvolatile function
- Applications that use the local transaction function
- Applications that use the global transaction function

For Web Application

- Applications that use the event channels nonvolatile function
- Applications that use the local transaction function

For EJB Application

It is possible to develop only EJB applications for sending messages. However, the following EJB applications can be developed according to intended usage:

- Applications that use the event channels nonvolatile function
- Applications that use the local transaction function
- Applications that use the global transaction function

Refer to Chapter 17, "Developing a JMS Application" for details.

Resource access definition

Make the resource access definition on the Interstage Management Console.

When Interstage client functions are installed, use the J2EE resource access definition. Refer to Chapter 19, "J2EE Resource Access Definition" for details.

Environment Setup when JavaMail is Referenced

When JavaMail is referenced, make the resource access definition on the Interstage Management Console.

When Interstage client functions are installed, use the J2EE resource access definition. Refer to Chapter 19, "J2EE Resource Access Definition" for details.

Environment Setup when URL is Referenced

When a URL is referenced, be sure to use the name conversion function.

Refer to "Name Conversion Function" for details of the name conversion function.

The following provides a description example of the name conversion file.

```
<?xml version="1.0"?>
<!DOCTYPE fujitsu-ebe-definition SYSTEM 'fujitsu-ebe-definition.dtd'>
<fujitsu-ebe-definition>
  <ejb>
    <group-name>MyServer</group-name>
    <jndi-name>OperationBean</jndi-name>
    <res-entry>
      <res-ref-name>url/SVURL</res-ref-name>
      <datasource-name>
        http://133.226.64.150/cgi-bin/CAmngtop.cgi
      </datasource-name>
    </res-entry>
  </ejb>
</fujitsu-ebe-definition>
```

Environment Setup when connector is Referenced

When a connector is referenced, deploy a resource adapter, then set the environment.

The following explains the resource adapter deployment and environment setting.

Deploying resource adapter

Use the Interstage Management Console to deploy the resource adapter in the operation environment.

For the deployment, specify an EAR file that contains a resource adapter file (rar file) or resource adapter file.

The following indicates the resource definition information items to be set for the deployment.

Refer to the resource adapter specification for details of the config-property information.

Setting item name	Setting contents
Resource adapter file name	The specified resource adapter file is displayed. The resource adapter file name cannot be changed on this window.
Resource name	Determine the resource name. The resource name is to be added to JNDI.
User name/password	Specify the user name and password used to connect to a resource. These values can be omitted, but do not specify the password only.
config-property information	Set this item name when you want to change the config-property information defined in the deployment descriptor. Only the property value can be changed.

The resource adapter file is extended in the following directory configuration. Refer to the resource adapter's manual and set CLASSPATH, PATH and LIBRARYPATH if required.

- Deploying in IJServer

  Refer to "IJServer file configuration" of "Environment Where J2EE Applications are Operated (IJServer)".
  ”jar” of RAR is set in the CLASSPATH automatically.

- To deploy, click [Resource] > [connector] in the Interstage Management Console.

  Windows

```
J2EE common directory\deployed\jca\ra\[Resource name]
*  The J2EE common directory default value is
C:\Interstage\J2EE\var\deployment.
```

  Solaris OE Linux

```
/opt/FJSVj2ee/var/deployment/deployed/jca/ra/[Resource name]
```

When distributed transaction is used

Select [Resource] > [connector] > [Deploy] on the Interstage Management Console and set "Use Global Transaction" to "Use".

Note

Distributed transaction cannot be used for deployment in IJServer. To deploy, click [Resource] > [connector] in the Interstage Management Console.

Referencing and changing resource definition

After the deployment ends, the resource adapter's information can be referenced by using the Interstage Management Console. The user name/password and config-property information property values set during the deployment can be changed

Environment setup

When the resource adapter is operated, the following environment setup is required after the deployment execution.

Because the RAR file is expanded during the deployment execution, set the environment variable as required. Refer to "Deploying resource adapter " for details of the deployed directory storage location.

The following example is to set PATH and CLASSPATH when the resource name is RA01, and the RAR file contains a library and RA01.jar.

- Select [WorkUnit] > "WorkUnit Name" > [WorkUnit Setting] and set the following in [Path] on the Interstage Management Console.

  Windows

  ```
  C:\Interstage\J2EE\var\deployment\deployed\jca\ra\RA01
  ```

  Solaris OE Linux

  ```
  /opt/FJSVj2ee/var/deployment/deployed/jca/ra/RA01
  ```

- Select [WorkUnit] > "WorkUnit Name" > [WorkUnit Setting] and set the following in [CLASSPATH] on the Interstage Management Console.

  Windows

  ```
  C:\Interstage\J2EE\var\deployment\deployed\jca\ra\RA01\RA01.jar
  ```

  Solaris OE Linux

  ```
  /opt/FJSVj2ee/var/deployment/deployed/jca/ra/RA01/RA01.jar
  ```

When distributed transaction is used

Windows

Set PATH and CLASSPATH in the system environment variable. Note that after the setting, the OS must be rebooted.

Solaris OE Linux

Set PATH and CLASSPATH in the system environment variable. Note that the setting must be made before Interstage is started up.

Description in deployment descriptor file

Write reference object information to the deployment descriptor file.

The following explains the tags for object reference.

Refer to the following for details of the deployment descriptor file.

- "J2EE Application Client deployment descriptor file Detailed Set Up" for J2EE application clients
- "Web application environment definition file (deployment descriptor)" in Chapter 6 for Web applications
- For EJB applications, use Apworks Apdesigner's EJB deployment descriptor editor or component designer's deployment descriptor file editor (*1).

  Refer to the “Apworks J Business Kit Online Manual” or “Component Designer User's Guide” for details. (*1).

  *1 "Component designer" is not provided by Plus Developer.

Write each object's information to the following deployment descriptor tag.

Deployment descriptor tag	**Specified value**
ejb-ref	EJB Home object
ejb-local-ref	EJB Local Home object
resource-ref	JDBC datasource
	JMS connection factory
	JavaMail mail session
	URL(Uniform Resource Locator)
	connector connection factory
resource-env-ref	JMS Destination
env-entry	Environment entry
Specification not required	UserTransaction
	ORB

Tag explanation

Tag		Explanation
ejb-ref		Defines the information related to EJB reference. It can be specified more than once.
	description	Specifies optional information to be indicated to the user. It can be omitted.
	ejb-ref-name	Specifies Enterprise Bean's reference name with the following prefix added. (xxxxx: optional character string) • ejb/xxxxx
	ejb-ref-type	Specifies Enterprise Bean's application type in one of the following. • Entity • Session
	home	Specifies a home interface name. As the home interface name, specify a restricted name (a package internal interface name).
	remote	Specifies a remote interface name. As the remote interface name, specify a restricted name (a package internal interface name).
	ejb-link	Specifies a Enterprise Bean name. It can be omitted.
ejb-local-ref		Defines information related to a local interface's EJB reference. It can be specified more than once.
	description	Specifies optional information to be indicated to the user. It can be omitted.
	ejb-ref-name	Specifies Enterprise Bean's reference name with the following prefix added. (xxxxx: optional character string) • ejb/xxxxx
	ejb-ref-type	Specifies Enterprise Bean's application type in one of the following. • Entity • Session
	local-home	Specifies a local home interface name. As the local home interface name, specify a restricted name (a package internal interface name).
	local	Specifies a local interface name. As the local interface name, specify a restricted name (a package internal interface name).
	ejb-link	Specifies a Enterprise Bean name. It can be omitted.

Tag		Explanation
resource-ref		Defines information related to a resource reference. It can be specified more than once.
	description	Specifies optional information to be indicated to the user. It can be omitted.
	res-ref-name	Specifies reference name with the following prefix added. (xxxxx: optional character string) • In the case of JDBC :jdbc/xxxxx • In the case of JMS :jms/xxxxx • In the case of JavaMail :mail/xxxxx • In the case of URL :url/xxxxx • In the case of connector :eis/xxxxx
	res-type	Use a restricted name to specify a type to be received by lookup. Specify an object class name or interface name in the restricted name. • In the case of JDBC :javax.sql.DataSource • In the case of JMS : javax.jms.TopicConnectionFactory or javax.jms.QueueConnectionFactory • In the case of JavaMail :javax.mail.Session • In the case of URL :java.net.URL • In the case of connector : javax.resource.cci.ConnectionFactory
	res-auth	Specifies a resource connector in one of the following. Application: Uses connection information set by application. Container: Uses connection information set by resource definition.
resource-env-ref		Defines information related to a resource environment reference. It can be specified more than once.
	description	Specifies optional information to be indicated to the user. It can be omitted.
	resource-env-ref-name	Specifies reference name with the following prefix added. (xxxxx: optional character string) • In the case of JMS :jms/xxxxx
	resource-env-ref-type	Use a restricted name to specify a type to be received by lookup. Specify an object class name or interface name in the restricted name. • In the case of JMS : javax.jms.Topic or javax.jms.Queue

Tag		Explanation
env-entry		Defines the information related to environment entry reference. It can be specified more than once.
	description	Specifies optional information to be indicated to the user. It can be omitted.
	env-entry-name	Specifies environment entry reference name .
	env-entry-type	Specifies environment entry type in one of the following. • java.lang.Boolean • java.lang.Byte • java.lang.String • java.lang.Short • java.lang.Integer • java.lang.Long • java.lang.Float • java.lang.Double • java.lang.Character
	env-entry-value	Specifies the environment entry value that the user wants to acquire by lookup. It can be omitted.

Setting example

[Setting example of EJB Home object “ejb/EJB1”]

```
<ejb-ref>
  <description>EJB Information</description>
  <ejb-ref-name>ejb/EJB1</ejb-ref-name>
  <ejb-ref-type>Session</ejb-ref-type>
  <home>sample.ejbHome</home>
  <remote>sample.ejbRemote</remote>
  <ejb-link>SessionBean</ejb-link>
</ejb-ref>
```

[Setting example of EJB Local Home object “ejb/SampleBMP”]

```
<ejb-local-ref>
    <ejb-ref-name>ejb/SampleBMP</ejb-ref-name>
    <ejb-ref-type>Entity</ejb-ref-type>
    <local-home>SampleBMPHome</local-home>
    <local>SampleBMPLocal</local>
    <ejb-link>SampleBMP</ejb-link>
</ejb-local-ref>
```

[Setting example of JDBC datasource “jdbc/DB1”]

```
<resource-ref>
  <description>JDBC Information</description>
  <res-ref-name>jdbc/DB1</res-ref-name>
  <res-type>javax.sql.DataSource</res-type>
  <res-auth>Application</res-auth>
</resource-ref>
```

[Setting example of JMS connection factory "jms/JMS1" and JMS destination "jms/JMS2"]

```
<resource-ref>
  <description>JMS Information</description>
  <res-ref-name>jms/JMS1</res-ref-name>
  <res-type>javax.jms.TopicConnectionFactory</res-type>
  <res-auth>Application</res-auth>
</resource-ref>
<resource-env-ref>
  <description>JMS Information2</description>
  <resource-env-ref-name>jms/JMS2</resource-env-ref-name>
  <resource-env-ref-type>javax.jms.Topic</resource-env-ref-type>
</resource-env-ref>
```

This setting example is for a J2EE application client deployment descriptor.

For Web and EJB applications, reverse the definition order of the following two tags, <resource-ref> and <resource-env-ref>.

[Setting example of JavaMail mail session “mail/Mail”]

```
<resource-ref>
  <res-ref-name>mail/Mail</res-ref-name>
  <res-type>javax.mail.Session</res-type>
  <res-auth>Application</res-auth>
</resource-ref>
```

[Setting example of URL “url/SVURL”]

```
<resource-ref>
  <res-ref-name>url/SVURL</res-ref-name>
  <res-type>java.net.URL</res-type>
  <res-auth>Application</res-auth>
</resource-ref>
```

[Setting example of connector connection factory “eis/RA01”]

```
<resource-ref>
  <res-ref-name>eis/RA01</res-ref-name>
  <res-type>javax.resource.cci.ConnectionFactory</res-type>
  <res-auth>Application</res-auth>
</resource-ref>
```

[Setting example of environment entry "SValue"]

```
<env-entry>
  <description>EnvProp</description>
  <env-entry-name>SValue</env-entry-name>
  <env-entry-type>java.lang.Short</env-entry-type>
  <env-entry-value>1024</env-entry-value>
</env-entry>
```

Notes

When the reference object information is not written to the deployment descriptor file, note the following.

- The container automatically retrieves objects of the same name. In this case, if different resources have objects of the same name, a malfunction may occur.
- The name conversion function cannot be used.
- A sub-context cannot be acquired.

Referencing Objects

Reference the objects in the following procedure.

1. Create a javax.naming.Context class object.
2. Using the lookup method, obtain the class object that suits the object to be referenced.

   Specify the following in the lookup method's argument.

   - In the case of EJB, “java:comp/env/ejb/EJB application name"
   - In the case of JDBC, “java:comp/env/jdbc/JDBC resource access definition name”
   - In the case of JMS, “java:comp/env/jms/JMS resource access definition name”
   - In the case of JavaMail, “java:comp/env/mail/JavaMail resource access definition name”
   - In the case of connector, “java:comp/env/eis/connector resource access definition name”
   - In the case of environment entry, “java:comp/env/environment entry name"
   - When the object is accessed using the name conversion, "java:comp/env/deployment descriptor's reference name"
3. When an EJB Home object is referenced, execute the narrow processing.

A description example is given below.

[Example of referencing EJB Home object whose EJB application name is EJB214ETY and Home class is EJB214ETYHome]

```
  //EJB Home object lookup processing
  java.lang.Object ejbobj = null;
  EJB214ETYHome home = null;
  try {
    javax.naming.Context nctx = new javax.naming.InitialContext();
    ejbobj = (java.lang.Object)nctx.lookup("java:comp/env/ejb/EJB214ETY");
    home = (EJB214ETYHome)javax.rmi.PortableRemoteObject.narrow(ejbobj,
EJB214ETYHome.class);
  } catch(javax.naming.NamingException ex) { }
```

Note:

When an Enterprise Bean's reference name is defined, the format called "ejb/Bean name should be used.

[Example of referencing EJB Local Home object whose EJB application name is EJB214EmpCBM and Local Home class is EJB214EmpCBMLocalHome]

```
  //EJB Local Home object lookup processing
  EJB214EmpCBMLocalHome home = null;
  try {
    javax.naming.Context nctx = new javax.naming.InitialContext();
    home =
(EJB214EmpCBMLocalHome)nctx.lookup("java:comp/env/ejb/EJB214EmpCBM");
  } catch(javax.naming.NamingException ex) { }
```

Note:

When an Enterprise Bean's reference name is defined, the format called "ejb/Bean name should be used.

[Example of referencing JDBC datasource whose JDBC resource access definition name is DB1]

```
  //JDBC datasource lookup processing
  javax.sql.DataSource dataSource = null;
  try {
    javax.naming.Context nctx = new javax.naming.InitialContext();
    dataSource = (javax.sql.DataSource)nctx.lookup("java:comp/env/jdbc/DB1");
  } catch(javax.naming.NamingException ex) { }
```

[Example of referencing JMS connection factory whose JMS resource access definition names are Topic and Queue]

[When JMS connection factory is javax.jms.TopicConnectionFactory]

```
  //JMS connection factory lookup processing
  javax.jms.TopicConnectionFactory topic = null;
  try {
    javax.naming.Context nctx = new javax.naming.InitialContext();
    topic =
(javax.jms.TopicConnectionFactory)nctx.lookup("java:comp/env/jms/Topic");
  } catch(javax.naming.NamingException ex) { }
```

[When JMS connection factory is javax.jms.QueueConnectionFactory]

```
  //JMS connection factory lookup processing
  javax.jms.QueueConnectionFactory queue = null;
  try {
    javax.naming.Context nctx = new javax.naming.InitialContext();
    queue =
(javax.jms.QueueConnectionFactory)nctx.lookup("java:comp/env/jms/Queue");
  } catch(javax.naming.NamingException ex) { }
```

[Example of referencing JavaMail mail session whose JavaMail resource access definition name is MailSession]

```
  //JavaMail mail session lookup processing
  javax.mail.Session session = null;
  try {
    javax.naming.Context nctx = new javax.naming.InitialContext();
    session =
(javax.mail.Session)nctx.lookup("java:comp/env/mail/MailSession");
  } catch(javax.naming.NamingException ex) { }
```

[Example of referencing URL whose deployment descriptor reference name is url/SVURL]

```
  //URL lookup processing
  java.net.URL url = null;
  try {
    javax.naming.Context nctx = new javax.naming.InitialContext();
    url = (java.net.URL)nctx.lookup("java:comp/env/url/SVURL");
  } catch(javax.naming.NamingException ex) { }
```

[Example of referencing connector connection factory whose connector resource access definition name is RA01]

```
  //connector connection factory lookup processing
  javax.resource.cci.ConnectionFactory cf = null;
  try {
    javax.naming.Context nctx = new javax.naming.InitialContext();
    cf
=(javax.resource.cci.ConnectionFactory)nctx.lookup("java:comp/env/eis/RA01")
;
  } catch(javax.naming.NamingException ex) { }
```

[Example of referencing JMS Destination whose JMS resource access definition names are Topic and Queue]

[When JMS Destination is Topic]

```
  //JMS Destination(javax.jms.Topic) lookup processing
  javax.jms.Topic topic = null;
  try {
    javax.naming.Context nctx = new javax.naming.InitialContext();
    topic = (javax.jms.Topic)nctx.lookup("java:comp/env/jms/Topic");
  } catch(javax.naming.NamingException ex) { }
```

[When JMS Destination is Queue]

```
//JMS Destination(javax.jms.Queue) lookup processing
javax.jms.Queue queue = null;
try {
  javax.naming.Context nctx = new javax.naming.InitialContext();
  queue = (javax.jms.Queue)nctx.lookup("java:comp/env/jms/Queue");
} catch(javax.naming.NamingException ex) { }
```

[Example of referencing environment entry whose environment entry name is SValue]

```
//environment entry lookup processing
java.lang.Short val = null;
try {
  javax.naming.Context nctx = new javax.naming.InitialContext();
  val = (java.lang.Short)nctx.lookup("java:comp/env/SValue");
} catch(javax.naming.NamingException ex) { }
```

Note

The lookup() method can be executed without "java:comp/env/" being specified in the argument; however, to do so is not recommended when application convertibility is emphasized.

[Example of referencing ORB object]

The ORB object that the container uses in RMI over IIOP communication can also be referenced as follows.

```
org.omg.CORBA.ORB orb = null;
try {
  javax.naming.Context nctx = new javax.naming.InitialContext();
  orb = (org.omg.CORBA.ORB)nctx.lookup("java:comp/ORB");
} catch(javax.naming.NamingException ex) { }
```

Name Conversion Function

This function maps the JNDI name to be specified in an application and the operation environment real name. Even if the JNDI name differs from the operation environment real name, using the name conversion function allows the user to respond without changing the application source JNDI name.

- For Web and EJB applications

  To set the name conversion, select [System] > [WorkUnit] > "WorkUnit Name" > "Module Name" > [Convert Name] tag on the Interstage Management Console for each module.

  The interstage.xml file can directly be edited. It is stored in the following position. Refer to "interstage.xml file" for details of the interstage.xml file.

  In addition, when a definition is changed while the WorkUnit is activated, the contents of the definition become effective by the following operation:

  Re-start of the WorkUnit.

  On the Interstage Management Console, choose a deployment module from [WorkUnit] > "WorkUnit Name" > [Application State/Undeploy] tab, and click the Reactivate button.

  Windows

  J2EE common directory\ijserver\[IJServer name]\apps\[module name]\META-INF\interstage.xml

  (The default J2EE common directory location is C:\Interstage\J2EE\var\deployment.)

  Solaris OE Linux

  /opt/FJSVj2ee/var/deployment/ijserver/[IJServer name]/apps/[module name]/META-INF/interstage.xml

- For J2EE application client

  Locate the name conversion file in the following directory and specify the file name in an environment property. Refer to "Name conversion file" for details of the name conversion file.

  Windows

  C:\Interstage\J2EE\etc

  Solaris OE

  /etc/opt/FJSVj2ee/etc

  When the system name (environment property: com.fujitsu.interstage.isas.SystemName) is specified in new javax.naming.InitialContext(Hashtable environment)'s argument environment or in application start time command line's argument (-D), the name conversion file must be stored in the following position.

  /var/opt/FJSVisas/system/[system name]/FJSVj2ee/etc/

Linux

/etc/opt/FJSVj2ee/etc

Note

When reference object information is not written to the deployment descriptor file, the name conversion function cannot be used.

Name conversion file

Description Format

The description format of the name conversion file is XML. The description format of the name conversion file is shown below.

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE fujitsu-ebe-definition SYSTEM 'fujitsu-ebe-definition.dtd'>
<fujitsu-ebe-definition>
  <client>
    <app-name>app-name</app-name>
    <ejb-ref-entry>
      <ejb-ref-name>ejb-ref-name</ejb-ref-name>
      <jndi-name>jndi-name</jndi-name>
    </ejb-ref-entry>
    <res-entry>
      <res-ref-name>res-ref-name</res-ref-name>
      <datasource-name>datasource-name</datasource-name>
    </res-entry>
    <res-env-entry>
      <res-env-ref-name>res-env-ref-name</res-env-ref-name>
      <environment-name>environment-name</environment-name>
    </res-env-entry>
  </client>
</fujitsu-ebe-definition>
```

- The first and second lines indicate the XML declaration and DTD (document type definition), which must be indicated at the beginning of the name conversion file.
  <fujitsu-ebe-definition> and </fujitsu-ebe-definition> on the third and last lines are the root tags that indicate the beginning and end of the XML file. Be sure to specify these tags.
- Indicate each tag in the above order.
- Bold character parts must be specified. **<app-name>** is required.
- Specify an optional character string in the italic character part. Control characters such as null, tab, and line feed characters cannot be used. Note that the italic character part is case-sensitive.

- When characters with special meanings (<, >, and &) are used in the XML file, write them according to the conversion definition as follows.

Character that you want to use	Notation in the name conversion file
<	<
>	>
&	&

- When "'" and """ are described in the character sequence of a value, it is interpreted as a single quotation mark (') and a double quotation mark (") respectively.

Tag explanation

Tag		Explanation
<client>		Specified for a J2EE application client. It can be specified more than once.
<app-name>		Specifies a name-converted application name.
<ejb-ref-entry>		Defines EJB object name conversion. It can be specified more than once. For one definition of this tag, define the following two tags one by one.
	<ejb-ref-name>	Specifies a deployment descriptor reference name.
	<jndi-name>	Specifies a EJB application name (operation environment real name) corresponding to <ejb-ref-name>.
<res-entry>		Defines JDBC data source, JMS (QueueConnectionFactory,TopicConnectionFactory), JavaMail, connector, and URL name conversion. It can be specified more than once. For one definition of this tag, define the following two tags one by one.
	<res-ref-name>	Specifies a deployment descriptor reference name.
	<datasource-name>	Specifies a resource access definition name (operation environment real name) corresponding to <res-ref-name>.
<res-env-entry>		Defines JMS Destination(Queue, Topic) name conversion. It can be specified more than once. For one definition of this tag, define the following two tags one by one.
	<res-env-ref-name>	Specifies a deployment descriptor reference name.
	<environment-name>	Specifies a resource access definition name (operation environment real name) corresponding to <res-env-ref-name>.

Description example (for J2EE application client)

A description example of a interstage.xml file applied when a deployment descriptor reference name and operation environment real name are as follows is given below.

	Deployment descriptor reference name	Operation environment real name
EJB	ejb/EntBean	EB1
Resource reference (JDBC)	jdbc/DataSource	DS1
Resource reference (JMS)	jms/TopicCF	CF1
Resource environment reference	jms/Topic	DN1

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE fujitsu-ebe-definition SYSTEM 'fujitsu-ebe-definition.dtd'>
<fujitsu-ebe-definition>
  <client>
    <app-name>GetBeans</app-name>
    <ejb-ref-entry>
      <ejb-ref-name>ejb/EntBean</ejb-ref-name>
      <jndi-name>EB1</jndi-name>
    </ejb-ref-entry>
    <res-entry>
      <res-ref-name>jdbc/DataSource</res-ref-name>
      <datasource-name>DS1</datasource-name>
    </res-entry>
    <res-entry>
      <res-ref-name>jms/TopicCF</res-ref-name>
      <datasource-name>CF1</datasource-name>
    </res-entry>
    <res-env-entry>
      <res-env-ref-name>jms/Topic</res-env-ref-name>
      <environment-name>DN1</environment-name>
    </res-env-entry>
  </client>
</fujitsu-ebe-definition>
```

interstage.xml file

Description Format

The description format of the interstage.xml file is XML. The description format of the interstage.xml file is shown below.

```
<?xml version="1.0"?>
<!DOCTYPE fujitsu-ebe-definition SYSTEM 'fujitsu-ebe-definition.dtd'>
<fujitsu-ebe-definition>
  <web>  or  <ejb>
    <group-name>group-name</group-name>
    <app-name>app-name</app-name>  or  <jndi-name>jndi-name</jndi-name>
    <ejb-ref-entry>
      <ejb-ref-name>ejb-ref-name</ejb-ref-name>
      <jndi-name>jndi-name</jndi-name>
    </ejb-ref-entry>
    <res-entry>
      <res-ref-name>res-ref-name</res-ref-name>
      <datasource-name>datasource-name</datasource-name>
    </res-entry>
    <res-env-entry>
      <res-env-ref-name>res-env-ref-name</res-env-ref-name>
      <environment-name>environment-name</environment-name>
    </res-env-entry>
  </web>  or  </ejb>
</fujitsu-ebe-definition>
```

- The first and second lines indicate the XML declaration and DTD (document type definition), which must be indicated at the beginning of the interstage.xml file.
- <fujitsu-ebe-definition> and </fujitsu-ebe-definition> on the third and last lines are the root tags that indicate the beginning and end of the XML file. Be sure to specify these tags.
- Indicate each tag in the above order.
- Bold character parts must be specified. In the EJB application, **<group-name>** and <jndi-name> tags are required. In the WEB application, **<app-name>** is required.
- Specify an optional character string in the italic character part. Control characters such as null, tab, and line feed characters cannot be used. Note that the italic character part is case-sensitive.
- When characters with special meanings (<, >, and &) are used in the XML file, write them according to the conversion definition as follows.

Character that you want to use	Notation in the interstage.xml file
<	<
>	>
&	&

- When "'" and """ are described in the character sequence of a value, it is interpreted as a single quotation mark (') and a double quotation mark (") respectively.
- Do not edit except <web> and the <ejb> tag.

Tag explanation

Tag		Explanation
<web>		Specified for a Web application. It can be specified more than once.
<ejb>		Specified for a EJB application. It can be specified more than once.
<app-name>		Specifies a name-converted application name in the case of a Web application.
<group-name>		Specifies an IJServer name. When this tag is omitted for an EJB application, name conversion information is not enabled.
<jndi-name>		Specifies a name-converted EJB application name in the case of a EJB application.
<ejb-ref-entry>		Defines EJB object name conversion. It can be specified more than once. For one definition of this tag, define the following two tags one by one.
	<ejb-ref-name>	Specifies a deployment descriptor reference name.
	<jndi-name>	Specifies a EJB application name (operation environment real name) corresponding to <ejb-ref-name>.
<res-entry>		Defines JDBC data source, JMS (QueueConnectionFactory,TopicConnectionFactory), JavaMail, connector, and URL name conversion. It can be specified more than once. For one definition of this tag, define the following two tags one by one.
	<res-ref-name>	Specifies a deployment descriptor reference name.
	<datasource-name>	Specifies a resource access definition name (operation environment real name) corresponding to <res-ref-name>.
<res-env-entry>		Defines JMS Destination(Queue, Topic) name conversion. It can be specified more than once. For one definition of this tag, define the following two tags one by one.
	<res-env-ref-name>	Specifies a deployment descriptor reference name.
	<environment-name>	Specifies a resource access definition name (operation environment real name) corresponding to <res-env-ref-name>.

Description example (for Web application)

A description example of a interstage.xml file applied when a deployment descriptor reference name and operation environment real name are as follows is given below.

	Deployment descriptor reference name	Operation environment real name
EJB	ejb/EntBean	EB1
Resource reference (JDBC)	jdbc/DataSource	DS1
Resource reference (JMS)	jms/TopicCF	CF1
Resource environment reference	jms/Topic	DN1

```
<?xml version="1.0"?>
<!DOCTYPE fujitsu-ebe-definition SYSTEM 'fujitsu-ebe-definition.dtd'>
<fujitsu-ebe-definition>
  <web>
    <app-name>GetBeans</app-name>
    <ejb-ref-entry>
      <ejb-ref-name>ejb/EntBean</ejb-ref-name>
      <jndi-name>EB1</jndi-name>
    </ejb-ref-entry>
    <res-entry>
      <res-ref-name>jdbc/DataSource</res-ref-name>
      <datasource-name>DS1</datasource-name>
    </res-entry>
    <res-entry>
      <res-ref-name>jms/TopicCF</res-ref-name>
      <datasource-name>CF1</datasource-name>
    </res-entry>
    <res-env-entry>
      <res-env-ref-name>jms/Topic</res-env-ref-name>
      <environment-name>DN1</environment-name>
    </res-env-entry>
  </web>
</fujitsu-ebe-definition>
```

Description example (for EJB application)

A description example of a interstage.xml file applied when a deployment descriptor reference name and operation environment real name are as follows is given below.

	Deployment descriptor reference name	Operation environment real name
EJB	ejb/CallBean	AccountBean
Resource reference (JMS)	jms/TopicCF	CatalogCF

[EJB application example]

```
  ...
  javax.naming.Context ic = new javax.naming.InitialContext();

  Object obj = (Object)ic.lookup("java:comp/env/ejb/CallBean");
  CallBeanHome beanHome =
      (CallBeanHome)javax.rmi.PortableRemoteObject.narrow(obj,
CallBeanHome.class);
  ...
  javax.jms.TopicConnectionFactory cf =

(javax.jms.TopicConnectionFactory)ic.lookup("java:comp/env/jms/TopicCF");
  ...
```

[Description example of interstage.xml file]

The following provides a description example of an interstage.xml file applied when name conversion is performed with IJServer's EJB application names, OperationBean and EmployeeBean. In the following example, the IJServer name is MyServer.

```
  <?xml version="1.0"?>
  <!DOCTYPE fujitsu-ebe-definition SYSTEM 'fujitsu-ebe-definition.dtd'>
  <fujitsu-ebe-definition >
    <ejb>
      <group-name>MyServer</group-name>
      <jndi-name>OperationBean</jndi-name>
      <ejb-ref-entry>
        <ejb-ref-name>ejb/CallBean</ejb-ref-name>
        <jndi-name>AccountBean</jndi-name>
      </ejb-ref-entry>
      <res-entry>
        <res-ref-name>jms/TopicCF</res-ref-name>
        <datasource-name>CatalogCF</datasource-name>
      </res-entry>
    </ejb>
    <ejb>
      <group-name>MyServer</group-name>
      <jndi-name>EmployeeBean</jndi-name>
      <ejb-ref-entry>
        <ejb-ref-name>ejb/CallBean</ejb-ref-name>
        <jndi-name>AccountBean</jndi-name>
      </ejb-ref-entry>
      <res-entry>
        <res-ref-name>jms/TopicCF</res-ref-name>
        <datasource-name>CatalogCF</datasource-name>
      </res-entry>
    </ejb>
  </fujitsu-ebe-definition>
```

Transaction Function using the UserTransaction Interface

An application uses the UserTransaction interface to control transaction. The following section explains how to develop applications. The explanation includes the processing flow of the transaction function that uses the UserTransaction interface.

Flow of the Transaction Function when the UserTransaction Interface is Used

Figure 3-1 shows the flow of the transaction function processing when the UserTransaction interface is used.

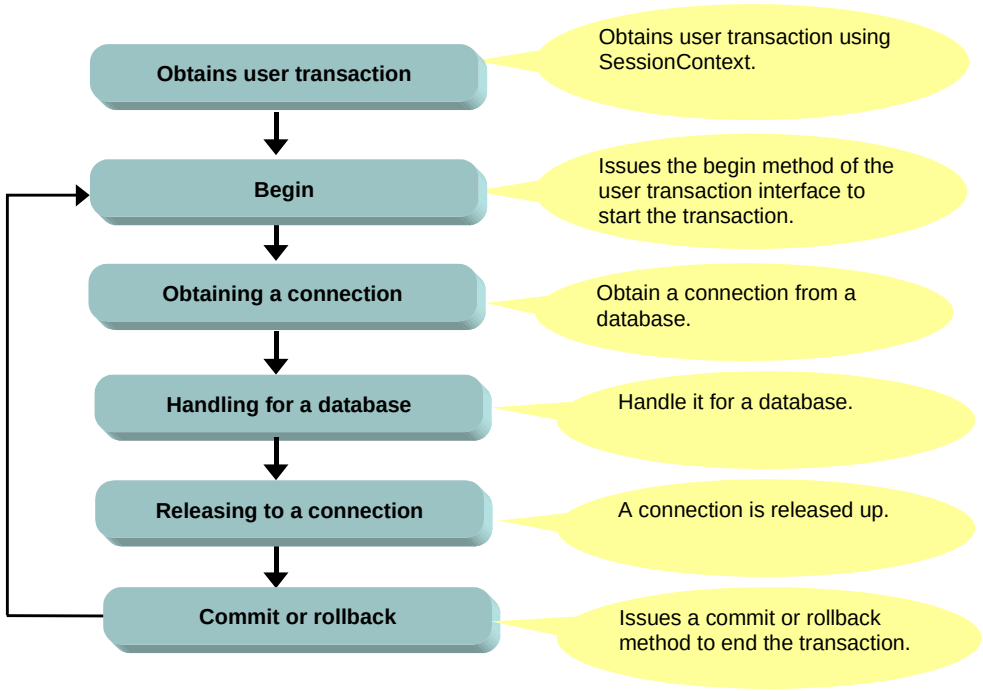


Figure 3-1 The Flow of Processing when the Transaction Function is Used

Methods of the UserTransaction Interface

The UserTransaction interface uses the following methods.

Table 3-1 shows the methods that can be used.

Table 3-1 UserTransaction Methods

Method Name	Description
begin	Creates a new transaction and associates it with the current thread.
commit	Completes a transaction associated with the current thread.
getStatus	Obtains the status of a transaction associated with the current thread.
rollback	Rolls back a transaction associated with the current thread.
setRollbackOnly	Changes a transaction associated with the current thread so that the transaction result is rollback only.
setTransactionTimeout	Changes the timeout associated with a transaction. Started by the begin method in the current thread.

Scope of transaction control

When the default transaction is started in a J2EE application, another J2EE application on the same JavaVM to be accessed can be operated in the same transaction.

When Web and EJB applications are operated on the same JavaVM, using UserTransaction, the Web application can transaction-link the EJB application processing accessed from that Web application.

Obtaining and Releasing a Connection

Issue the getConnection() method to a Datasource in order to obtain a connection. A connection obtained in any other way is not handled as the connection in a transaction.

Issue the close() method to an obtained connection in order to release it.

To use a Datasource, perform lookup on the Datasource.

Example

```
  ...
  javax.transaction.UserTransaction userTransaction = null ;
    /* UserTransaction is acquired by using JNDI lookup method.   */
    try {
        javax.naming.Context initialContext = new
        javax.naming.InitialContext();
        userTransaction =
        (UserTransaction)initialContext.lookup("java:comp/UserTransaction");
    } catch(NamingException ex) {
        /* Exception processing */
        ...
    }

    /* Transaction processing is started.  */
    try {
        userTransaction.begin();
    } catch(javax.transaction.NotSupportedException e) {
        /* Exception processing */
        ...
    } catch(javax.transaction.SystemException e) {
        /* Exception processing */
        ...
    }

    try {
        /* EJB application invocation or JDBC data source access  */
        ...
    } catch( Throwable e ) {
        /* If an exception occurs, the transaction is rolled back. */
        userTransaction.rollback();
        throw e;
    }

    try {
        /* When the user wants to complete the processing, the transaction is
committed.  */
        userTransaction.commit();
    } catch( Throwable e ) {
        throw e;
    }
  ...
```

J2EE Application Client deployment descriptor file Detailed Set Up

This section explains the coding format of the deployment descriptor file.

The name of the deployment descriptor is arbitrary and the extension is .xml.

Place the deployment descriptor file in any directory and specify the filename with an environment property using the full pathname.

Description Format

The description format of the deployment descriptor is XML. The description format of the deployment descriptor is shown below.

```
<?xml version="1.0" encoding="UTF-8"?>
  <!DOCTYPE application-client PUBLIC
  "-//Sun Microsystems, Inc.//DTD J2EE Application Client 1.3//EN"
  "http://java.sun.com/dtd/application-client_1_3.dtd">
  <application-client>
    <icon>
      <small-icon>small_icon</small-icon>
      <large-icon>large_icon</large-icon>
    </icon>
    <display-name>display_name</display-name>
    <description>description</description>
    <env-entry>
      <description>description</description>
      <env-entry-name>name</env-entry-name>
      <env-entry-type>type</env-entry-type>
      <env-entry-value>value</env-entry-value>
    </env-entry>
    <ejb-ref>
      <description>description</description>
      <ejb-ref-name>name</ejb-ref-name>
      <ejb-ref-type>type</ejb-ref-type>
      <home>home</home>
      <remote>remote</remote>
      <ejb-link>link</ejb-link>
    </ejb-ref>
    <resource-ref>
      <description>description</description>
      <res-ref-name>name<res-ref-name>
      <res-type>type</res-type>
      <res-auth>auth</res-auth>
    </resource-ref>
```

```
<resource-env-ref>
    <description>description</description>
    <resource-env-ref-name>name</resource-env-ref-name>
    <resource-env-ref-type>type</resource-env-ref-type>
  </resource-env-ref>
</application-client>
```

- The first and second lines indicate the XML declaration and DTD (document type definition), which must be indicated at the beginning of the name conversion file.
  When the Japanese language is used in value strings , specify appropriate values such as "Shift_JIS" in the encoding format ("encoding=" part).
- <application-client> and </application-client> on the third and last lines are the root tags that indicate the beginning and end of the XML file. Be sure to specify these tags.
- Indicate each tag in the above order.
- A tag character string is case-sensitive.
- Specify an optional character string in the italic character part. Control characters such as a null, tab, and line feed cannot be used. Note that the italic character part is case-sensitive.

Tag explanation

Tag		Explanation
icon		
	small-icon	Specify the URI to the small (16 X 16) icon (GIF/JPEG format) representing the J2EE application on the GUI. Specify the URI using the relative path from the package route.
	large-icon	Specify the URI to the large (32 X 32) icon (GIF/JPEG format) representing the J2EE application on the GUI. Specify the URI using the relative path from the package route.
display-name		Specify the J2EE application client display name. The J2EE application client display name is displayed, for example, in the GUI.
description		Specify detailed information about the J2EE application client. As the detailed information, specify any information that should be communicated to the user.

Refer to "Description in deployment descriptor file" for details of the following tags related to object reference.

- env-entry
- ejb-ref
- resource-ref
- resource-env-ref

Chapter 4

The J2EE Application Security Function

This chapter describes the J2EE application security function and covers the following issues related to J2EE application security.

- The Security Function

  This section explains the functions and details of the security function.

- Embedding the Security Function

  This section explains how to set up the security function.

- Collecting the Authentication Log of the Security Function

  This section explains how to collect the authentication log of the security function and the message format.

- Action when a Security Function Error Occurs

  This section explains the actions to be taken when an error in the security function occurs.

The Security Function

The security function prevents invalid access to the J2EE application resources.

Connections for operation

The J2EE application security function assumes the following connections for operation:

- J2EE application client to EJB application
- J2EE application client to EJB application to EJB application
- Web application to EJB application
- Web application to EJB application to EJB application

Types of security function

The security function in the J2EE application client provides the following:

- User authentication
- Access constraints
- Method permissions
- Security methods
- Resource-connectable user control function
- Run-as security function

User Authentication

About User Authentication

User authentication is a process to check for valid users using user IDs and passwords. User authentication prevents any access from invalid users.

Security roles

Security roles are permissions provided to users. Security roles are used to create groups of users, such as Administrator, Guest and Manager groups.

Security roles can be set through user authentication.

With security roles, access permissions can be specified for user groups.

For example, “allow access from users to which the Administrator or Manager security role is assigned” is possible.

Operation of other security functions

User authentication is a process to check for valid users.

The other security functions enable information to be obtained about each user through user authentication (user name, password, and security role) and allow access by the user within the permission granted for the user.

Directory Service

The Interstage Application Server uses Smart Repository (from now on referred to as the Directory Service) as a user/security role management register.

To use the security functions of J2EE applications, set up Directory Service and register the users.

Note

Interstage Application Server Web-J Edition does not include Smart Repository. This must be ordered and set up separately.

In this document, Solaris(TM) Operating Environment is from now on abbreviated as Solaris OE.

Applications which Authenticate Users

J2EE application client

The J2EE application client authenticates users when the user IDs and their passwords, which were specified in the JNDI environment property, have been set up in Directory Service.

Web applications

Web applications authenticate users when the user IDs and their passwords, which were input during the authentication dialogs, have been set up in Directory Service.

For user authentication, either of the following methods is allowed:

- HTTP Basic authentication

  To use authentication dialogs provided by the Web browser.

- Form based login

  To use specific pages (based on HTML or JSP) created as authentication dialog pages.

If Tomcat4.1 is changed to the Servlet service, it is recommended to change the user/security role management register to the Directory Service. Doing so allows the J2EE application to use the security function.

For Tomcat4.1 compatibility, the same realm tag as that of Tomcat4.1 can be set in the following server.xml definition file. With this setting, a security function using a management register other than the Directory Service can be used but the security function is closed within Web applications.

J2EE common directory\ijserver\IJServer WorkUnit name\server.xml

/var/opt/FJSVj2ee/deployment/ijserver/IJServer WorkUnit name/server.xml

Access Constraints

Access constraints can be placed on individual Web application resource based on:

- Security role
- Transport method

According to the result of a check on access constraints, the Servlet container returns one of the following responses to the Web browser via the Web server:

- Access allowed: HTTP status code 200
- Access prohibited/rejected user: HTTP status code 401
- Access prohibited/rejected transport method or security role: HTTP status code 403

Security Role

Access is restricted according to the security role obtained through user authentication.

Transport method

Access is restricted according to the transport method between the client and Web server.

The following transport method options are provided to restrict access:

- NONE: Does not require any assurance of data transport.
- INTEGRAL: Requires assurance of data transport.
- CONFIDENTIAL: Requires prevention of data tapping.

INTEGRAL and CONFIDENTIAL allow access with SSL used. For example "allow access with SSL enabled" is possible.

Method Permissions

Method permissions restrict access to EJB application methods.

A method permission works as follows:

1. There must be accessible security roles defined in an EJB application method.
2. When a user requests access to the EJB application method, the container obtains the security role from the user ID.
3. If the obtained security role has been defined in the method, access by the user is permitted.

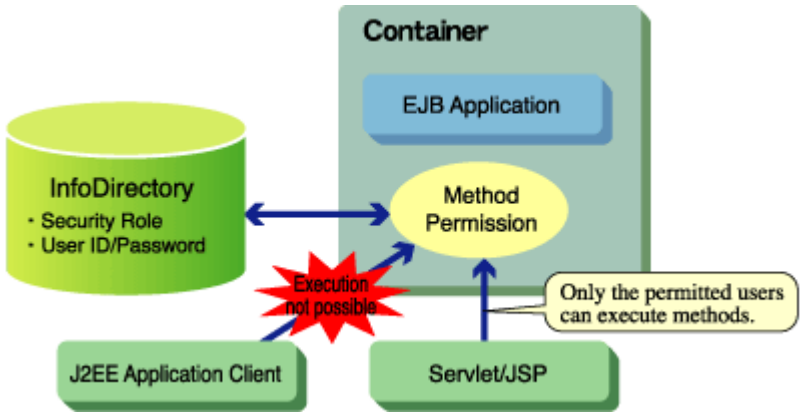


Figure 4-1 Method Permission

Method permissions use information about users, therefore the J2EE application client or a Web application must authenticate the user.

Security Methods

EJB applications can support the following security methods (javax.ejb.EJBContext interface method):

- getCallerPrincipal()
- isCallerInRole(java.lang.String roleName)

With these methods, it becomes possible to obtain information about authentication in EJB application business methods and permit access.

Resource-connectable User Control Function

The function for management of users with access to resources specifies users who are allowed to access resources to prevent invalid access to resources.

The function for management of users with access to resources is valid only when JDBC is assigned as the resource manager.

Define resource accessible users using "resource accessible user specification" (the res-auth tag in the resource-ref tag) of the deployment descriptor in the corresponding J2EE application.

The following values can be specified:

- Container: Uses access information that has been specified in the resource definition.
- Application: Uses access information that has been set in the application.

About Specification of Container

The connection information that is set in the resource definition includes the user ID and password specified on the Interstage Management Console.

About Specification of Applications

Access information can be set in an application with either of the following methods:

- When a user accesses a resource from an EJB application, and the user has been specified in the EJB application

  User ID and password specified in the EJB application.

- In other cases

  User ID and password of the user authenticated by the J2EE application client.

Note

Specification of applications is not supported in Web applications. The application operates as specified in the container.

Example

The following shows an example of specifying the resource connector in the J2EE application client, and defines the use of the user ID and password user-authenticated by the J2EE application client to access the resource called jdbc/DB1.

```
<resource-ref>
    <description>JDBC Information</description>
    <res-ref-name>jdbc/DB1</res-ref-name>
    <res-type>javax.sql.DataSource</res-type>
    <res-auth>Application</res-auth>
</resource-ref>
```

Run-as Security Function

The run-as security function is a function that can specify the authentication information for the EJB application.

Authentication information (security role, user ID, and password) used in normal EJB applications is information authenticated by the client (J2EE application client and Web application). By using the run-as security function however, authentication information used by the EJB application can be changed.

This function is enabled when an EJB application, in which the security function as shown below is used, is accessed from Message-driven Bean:

- EJB application to which method permission is set
- EJB application in which "Application" is set to the resource connector management function

Because Message-driven Bean is not normally accessed from the client directly, it does not have authentication information. Thus, if an attempt is made to access an EJB application as above, an error is always caused by the security check. In that case, by setting authentication information to Message-driven Bean using the run-as security function, the access controlled EJB application can be accessed from the Message-driven Bean.

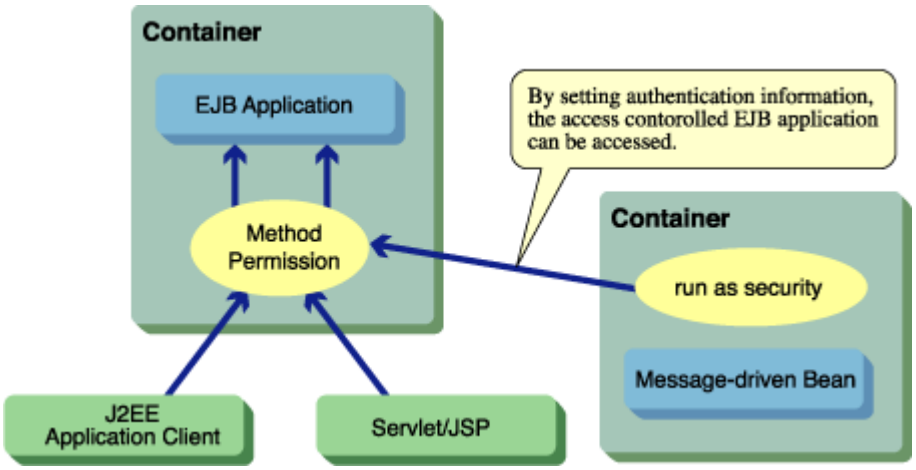


Figure 4-2 Run-as Security Function

The following settings are needed to use the run-as security function:

- deployment descriptor setting
- user ID/password setting
- Directory Service setting

Deployment Descriptor Setting

The run-as security function can be specified in the deployment descriptor. The security role name needs to be specified in the run-as tag of security-identity. By making this setting, the EJB application operates with the specified security role.

The deployment descriptor can be specified and changed by using Apworks or the Interstage Management Console.

Coding example

The following shows a coding example of the deployment descriptor of run-as security, where the security role called "Admin" is specified for the EJB application.

```
…
    <enterprise-beans>
        <security-identity>
            <run-as>
                <role-name>Admin</role-name>
            </run-as>
        </security-identity>
    </enterprise-beans>
…
```

User ID and Password Setting

Specify the user ID/password corresponding to the security role name specified in the run-as tag on the Interstage Management Console. In the following cases, a warning message EJB1078 is output at startup. If the warning message is output and an EJB application using the method permission function is called, an authorization error occurs.

- If the password is incorrect
- If the security role is not registered in Directory Service
- If the specified user ID is not registered with the specified security role

Directory Service Setting

For details, see the description of Directory Service Setting.

Notes

Consider the following points when using the run-as security function.

Use this function only for EJB applications that are called from outside IJServer.

If the function is used by EJB applications that are called from other EJB applications deployed to IJServer, the following operations may malfunction.

- Method permissions
- Resource-connectable user control function

Embedding the Security Function

This section describes how to install the security functions.

1. Directory Service setting
2. Setting for each application

Directory Service Setting

The setting procedure of Directory Service is explained as follows.

1. Setting Up the Security Management Environment Definition Files
2. User and Security Role Settings

Setting Up the Security Management Environment Definition Files

Define the operation environment of Directory Service in the security management environment definition files.

The Relationship between the security management environment definition file and Directory Service

Figure 4-3 (on the following page) shows the relationship between the security management environment definition file and Directory Service.

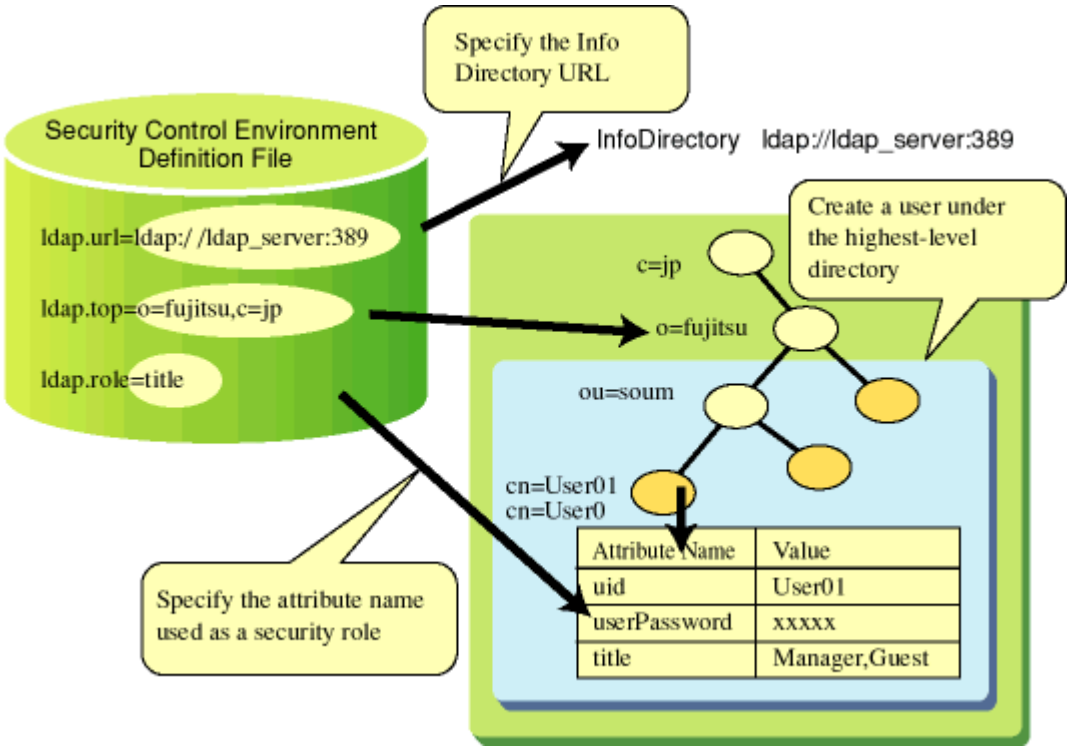


Figure 4-3 Relationship between the Security Management Environment Definition File and InfoDirectory

The filename of security management environment definition files

Installation of the J2EE package sets the security management environment definition files with the following filenames:

Windows

C:\Interstage\J2EE\etc\security.properties

Solaris OE Linux

/etc/opt/FJSVj2ee/etc/security.properties

Set up the security management environment definition file in the server environment and in the client environment respectively.

In the client environment, a security management environment definition file appropriate for the environment is installed. Set up the security management environment definition file in each client environment.

Be sure to set the same values for the items of the security management environment definition file in the server environment and those of the security management environment definition files in the client environments.

Security Management Environment Definition File Settings

The items to be set in each environment definition file are shown in Table 4-1 below.

Table 4-1 Security Management Environment Definition File Settings

Item	Setting	Default value
ldap.url	Specify the server URL of Directory Service . Specify it in the format "ldap://host name:port number".	Required
ldap.top	Specify the DN name of the top directory that stores the user. Specify the DN name starting from the top entry DN specified for Directory Service . The user created under the top directory can be used in the security function.	Required
ldap.role	Specify the attribute name of the user object used as a security role.	Required

Edit the security management environment definition file using a text editor.

Specify each item in the format "Item name=setting" in one line. If an item name that does not exist is specified, the line is considered to be a comment line.

Example

```
ldap.url=ldap://ldap_server:389
ldap.top=o=fujitsu,c=jp
ldap.role=title
```

When any definition item in the security management environment definition file is changed, restart each container.

User and Security Role Settings

To define users, set up the user IDs, passwords, and security roles for the users to be handled by the security functions in the directory in which Directory Service was created.

To define the users, use the management tools provided by Directory Service .

Define the users using the following object class:

Set the attribute values given in Table 4-2 to the user ID, password, and security role.

Table 4-2 Security Function Attribute Values

Attribute name	Attribute value	Remarks
uid	User ID	Set a user ID unique under the top directory (Define the top directory in the security management environment definition file). If the user ID is duplicated, the security cannot be applied correctly.
userPassword	Password	2-byte code characters cannot be specified.
securityRole	Security role	When specifying multiple security roles, separate them by a comma ','. 2-byte code characters cannot be specified.

Names of attributes for security roles

Define names of attributes to be used for security roles in the security management environment definition file. Set the security roles to existing Directory Service attributes.

For example, when "ldap.role=title" is defined in the security management environment definition file, the value specified for the attribute "title" is handled as a security role.

Delay for changes to take effect

Any access to Directory Service is cached.

Therefore, changes or deletions of passwords, security roles, or users in Directory Service may not take effect before 30 minutes have elapsed, and the cache stores information about access during this time.

To make changes or deletions effective immediately, restart all functions that use the security functions.

Directory Service work procedure

This section explains the work procedure for using the security function of the J2EE application using the management tools provided by the Interstage Management Console and Directory Service. The Smart Repository work procedures are shown below.

Smart Repository Service work procedure

Refer to the Smart Repository Operator's Guide for details of the Smart Repository function.

[Operation with Interstage Management Console]

1. Create a repository.
2. Start the repository.

[Operation with Entry Administration Tool]

3. Set the connection repository.
4. Log in to the repository.
5. Register the user.

1. Creating a repository

Create a repository using the Interstage Management Console.

1. Start the Interstage Management Console.
2. Select [System] > [Service] > [Repository] > [Create New] tab, and enter the password of the administrator DN to create a repository. Use the default values for the input items other than the administrator DN password.

2. Starting the repository

Start the repository from the Interstage Management Console.

Select [System] > [Service] > [Repository] and select the checkbox for the repository created in 1, then click the Start button.

3. Setting the connection repository

Set the connection repository using the Entry Administration Tool.

Windows

From the Windows(R) [Start] menu, select [Programs] - [Interstage] - [Application

Server] - [Smart Repository] - [Entry Administration Tool].

Solaris OE Linux

In the X window operating environment, enter the /opt/FJSVirep/gui/bin/irepeditent command to display the Entry Administration Tool.

1. Select [Connect] - [Set for Connection] from the [Entry Administration Tool] window.

   -> The [Connection List] window appears.

2. Click the [New location for connection] button.

   -> The connection name input window appears.

3. Enter the connection name and click the OK button. In the [Connection List] window, enter the host name of the created repository, port, public directory, and administrator DN, and click the Save button.

   -> A confirmation dialog box appears.

4. Click the OK button, and then click the Close button.

4. Logging in to the repository

Log in to the repository with repository administrator authority.

1. Select [Connect] - [Login] from the [Entry Administration Tool] window.

   -> The connection name and password input dialog box appears.

2. Select the connection name and enter the administrator DN password, then click the Login button.

5. Registering the user

Register the user as an entry.

1. Double-click the top entry displayed under "Directory" in the [Entry Administration Tool] window. The top entry is the same as the setting of 1dap.url in the security management environment definition file (security.properties).

   -> Organization unit "User" is displayed.
   * Organization unit "User" is created as the default tree when a repository is created.
   If organization unit "User" is not displayed, check the setting for creating a default tree during repository creation.
2. Select organization unit "User" as a user registration entry. In this case the setting in the security management environment definition file (security.properties) is as shown below

   ldap.top=ou=User,ou=interstage,o=fujitsu,dc=com
3. While selecting the user registration entry, right-click and select [Add].
4. Select the Internet user from the [List of object class] panel.
5. Enter the following types of information in the right frame:

cn	Unique name that identifies the user (such as an employee number)
sn	Family name of the user
givenName	Given name of the user
uid	Login name used for authentication
employeeNumber	Employee number
userPassword	Password used for authentication
ou	Organization unit to which the user belongs

6. Click the [Add Attribute] button and enter role name information.

Attribute name	**Attribute value**
title	Enter the role name corresponding to the user.

   * If the schema name that is set in "ldap.role" in the security management environment definition file (security.properties) is changed from "title," another schema name can also be used as the save destination.
7. Click the [OK] button.

   -> The relevant user is added to the selected entry.

Setting the Security Function into the J2EE Application Client

Setting up the User Authentication

Setting method

To set up user authentication on the J2EE application client, specify the following JNDI environment property settings:

- FJUserID: Specifies a user ID to be used for user authentication in Directory Service.
- FJPassword: Specifies a password to be used for user authentication in Directory Service.

Set up FJUserID and FJPassword using one of the following:

- FJjndi.properties file
- environment argument for new javax.naming.InitialContext (Hashtable environment)
- Arguments(-D) in the command line at startup of the application

If a duplicate environment property is specified, it is overwritten with the following priority ("3" indicates the highest priority).

(1) FJjndi.properties file

(2) javax.naming.InitialContext (Hashtable environment) argument

(3) Argument in the command line at startup of the application (-D)

Setting example

The examples below show a sample setting of the JNDI environment property.

Setting specified with the FJjndi.properties file

```
FJUserID=user01
FJPassword=pass01
com.fujitsu.interstage.j2ee.DeploymentDescriptorClient=/export/home/j2eeapl/
application-client.xml
```

Setting specified with the new InitialContext argument

```
…
Context ctx = null;
try {
  Hashtable env = new Hashtable (5);
  env.put (“java.naming.factory.initial”,
    "com.fujitsu.interstage.j2ee.jndi.InitialContextFactoryForClient");
  env.put ("FJUserID", "user01");
  env.put ("FJPassword", "pass01");
  env.put (“com.fujitsu.interstage.j2ee.DeploymentDescriptorClient”,
    "/export/home/j2eeapl/application-client.xml");
  ctx = new InitialContext(env);
}
catch (NamingException ne) {
  ne.printStackTrace();
}
…
```

Setting specified with arguments in the command line at startup of the application

```
java -Djava.naming.factory.initial=
com.fujitsu.interstage.j2ee.jndi.InitialContextFactoryForClient
-Dcom.fujitsu.interstage.j2ee.DeploymentDescriptorClient=
/export/home/j2eeapl/application-client.xml -DFJUserID=user01
-DFJPassword=pass01 ClientAPP
```

Setting up the Resource-connectable User Control Function

See the Resource-connectable User Control Function.

Setting the Security Function into a Web Application

To use the security function in a Web application, specify the settings as follows.

Setting up the User Authentication

To use the user authentication on a Web application, set up login-config tag of the Web application environment definition file.

Setting up the Access Constraint

To use the access constraint on a Web application, set up following tags of the Web application environment definition file:

- security-constraint tag
- security-role tag
- security-role-ref tag

Sample setting

The example below shows a sample setting of the Web application environment definition file.

```
…
<servlet>

    <security-role-ref>
        <role-name>
            ADM
        </role-name>
        <role-link>
            Administrator →(*1)
        </role-link>
    </security-role-ref>

</servlet>

<security-constraint>
    <web-resource-collection>
        <web-resource-name>
            Shop
        </web-resource-name>
        <url-pattern>
            /Shop/* →(*2)
        </url-pattern>
        <http-method>
            POST →(*3)
        </http-method>
    </web-resource-collection>
    <auth-constraint>
        <role-name>
```

```
            Administrator →(*4)
        </role-name>
    </auth-constraint>
    <user-data-constraint>
        <transport-guarantee>
            CONFIDENTIAL →(*5)
        </transport-guarantee>
    </user-data-constraint>
</security-constraint>

<login-config>
    <auth-method>
        BASIC →(*6)
    </auth-method>
    <realm-name>
        name
    </realm-name>
</login-config>

<security-role>
    <role-name>
        Administrator →(*7)
    </role-name>
</security-role>

…
```

When access to the resource indicated by (*2) is made with the method indicated by (*3), HTTP Basic authorization is handled as indicated by (*6).

In the sample settings, security is specified only to Administrator as indicated by (*4).

The value indicated by (*4) must be also defined by (*1). In addition, the value indicated by (*1) must be also defined by (*7).

For the value in (*7), specify the value that is set for the title attribute of the Directory Service used as the security role.

For form based login, specify the settings as follows:

```
<login-config>
    <auth-method>
        FORM
    </auth-method>
    <form-login-config>
        <form-login-page>
            /login.jsp
        </form-login-page>
        <form-error-page>
            /error.jsp
        </form-error-page>
    </form-login-config>
</login-config>
```

Setting up the Resource-connectable User Control Function

See the Resource-connectable User Control Function.

Setting the Security Function into the EJB Application

Setting up the Method Permission

To use method permissions, set up in the following deployment descriptors:

- Security role name
- Security Reference
- Method Permission

For setting details, refer to the Apworks Apdesigner Programmer's Guide or ComponentDesigner User's Guide (Note: Not distributed for Plus Developer).

An Interstage Management Console can also define a method permission.

Setting up the Resource-connectable User Control Function

Refer to the Resource-connectable User Control Function.

Collecting the Authentication Log of the Security Function

Using the Trace Function, authentication logs of the security function can be collected. This function allows you to check for illegal access.

Note

If either of the following conditions apply, the log is not output:

- If either the user ID or password is not specified
- If a blank character is specified as the user ID or password.

Setting method

Specify the following as JavaVM start-time parameter.

- Log filename

  -Dcom.fujitsu.interstage.j2ee.security.logfile = Log filename

- Log size

  -Dcom.fujitsu.interstage.j2ee.security.logsize = Log size

Specify the J2EE application client with the argument on the application start-time command line.

Specify Web application and EJB applications with the Java Command Options in the IJServer WorkUnit definition.

Explanation of the specification of the log filename and the log size is given in Table 4-3.

Table 4-3 Log Specification Items

Specification item	Log filename
Log filename	Specify the filename of the log. If it is specified with a relative path, it is interpreted as a relative path from the current directory on which JavaVM is running. If the log filename is not specified or if a directory name is specified as a log filename, logs are not collected. **Note** Specify a different log filename for each JavaVM to be started. If the same file is specified, logs output from two or more JavaVMs may be mixed or partial loss may occur.

Specification item	Log filename
Log size	Specify the maximum size of log information in MB. If the specified size is exceeded, a backup file is created with the following name, and a backup of the log is stored. (log-file-name).old The values that can be specified range from 1 to 2147483647. If a value other than a numeric value is specified or if this item is omitted, the log size is 1MB.

Message format

A log is output in the following format:

```
[day/month/year hour:minute:second] authentication trace (authentication method,
authentication result, reason for rejection) uid="user-name" role="role"
```

The following elements are output to each item.

Table 4-4 Log Output Items

Item	Description
[day/month/year hour:minute:second]	Day and time at which the event has occurred.
authentication trace	Identifier that specifies that it is access trace in security authentication.
authentication method	Displays the method in which the authentication has succeeded or is rejected. ldap: Authentication with Directory Service cache: Authentication with cache
authentication result	Displays the result of authentication. For authentication result, either of the following is displayed. true: Authentication succeeded false: Authentication rejected
Reason for rejection	When the authentication is rejected, its reason is displayed. The reasons are the following three cases. no data: No data exists in the cache. different password: The password is different. over time: The cache valid period has expired.
User name	Displays the user who is authenticated.
Role	If authentication has succeeded, the role corresponding to the user is displayed. If authentication is rejected, this item is not displayed.

An output example is shown below:

```
[09/01/2001 12:00:00.000] authentication trace (ldap, true, -) uid="Fujitsu"
role="Administrator"
[09/01/2001 12:01:01.000] authentication trace (ldap, false, no data)
uid="Fujitsu"
```

Action when a Security Function Error Occurs

If an error in the security function has occurred, an error message for each application will be sent to the following output media respectively.

- **J2EE application client**

  Logs of standard output/standard error output

- **Web application, EJB application**

  Container logs of the IJServer

When dealing with errors, refer to the following information as required:

- For details of security management environment definition files, refer to Setting Up the Security Management Environment Definition Files.
- For details of Smart Repository errors, refer to Messages Beginning with ‘irep’, or Messages Output by Smart Repository in the Interstage Messages manual.
- For details of trace function security function, refer to Collecting the Authentication Log of the Security Function.

For details of error messages, refer to Messages Output by J2EE Application Security Function in the Interstage Messages manual.

Part II

Servlet/JSP Edition

Chapter 5

Functions of the Servlet Service

This chapter describes the functions of the Servlet Service.

Input Code Automatic Conversion Function

The input code automatic conversion function performs code conversion of a request parameter from a Web client.

If this function is used, code conversion need not be performed in an application. Therefore, you can change the code to be handled without modifying a program but by simply changing the environment definition. Additionally, the code to be handled can be specified for each Web application.

If "JISAutoDetect" (automatic modification) is specified, the character set of short Japanese character strings may not be recognized correctly and character set modification may not be performed correctly.

Use the Interstage Management Console.

Custom Tag Pooling Function

The Servlet container provides the function to pool custom tag objects used by JSP.

This function allows reusing of existing custom tag objects and enhances the process response.

Use the Interstage Management Console to enable or disable the custom tag pooling function for each IJServer.

Note

The custom tag pooling function calls the following method to reuse a custom tag object:

```
javax.servlet.jsp.tagext.Tag interface release method
```

The Java Server Pages (TM) 1.2 specification indicates that the method must be correctly installed.

Even custom tags that are not compliant with this specification can operate normally if "disable the custom tag pooling function" is selected.

Chapter 6

Web Application Development

Web applications consist of Web resources such as HTML files, image files, servlets and JSP files, and Web application environment definition files. It is possible to develop functionality as a single Web application package.

This chapter explains:

Notes on the Development of Web Applications

Web Application Environment Definition File (Deployment Descriptor)

Refer to "Debugging Application" in Chapter 2 for details of debugging.

Notes on the Development of Web Applications

This chapter describes the points to be noted when developing Web applications.

Notes when Using Cookies

In some Web browsers, the event when the port number is specified to 80 (in the case of the SSL communication, the port number is 443), and the event when the port number is not specified, are judged to be different servers. As a result, it is possible that the Cookie header is not transmitted. To prevent this problem, it is recommended to use HTML or construct applications so that the method of calling from the Web browser is unified to either one of them when application control is going to be performed using a Cookie.

Cross-site-scripting Fragility Problem

An application that returns the input values directly to the browser or returns the contents of a Java error or exception may become a security hole (because of the vulnerability of Cross-site-Scripting).

It is recommended not to create such applications.

For information on Cross-site-Scripting, refer to "About the Cross-Site Scripting Problem" in "Common notes on Interstage" in the Product Notes.

Errors and Exceptions

Fujitsu recommends not using an application that returns to the browser an error or exception that occurred in the application. This usage may lead to leakage of internal information.

If the servlet or JSP has not processed (caught) an error or an exception that has occurred, and the error page is not specified in the web application environment definition file or JSP, the error page held by the servlet container is displayed. In this case, exception and error stack trace data is not output.

Specifying an error page for the HTTP error status code

This section explains the location for specifying the error page for the HTTP error status code and the error page that is used.

Location for specifying the error page for the HTTP error status code

The locations for specifying the error page for the HTTP error status code are as follows:

Web application environment definition file (deployment descriptor)

Web server environment settings

The location for specifying the error page depends on where the problem occurred. Change or unify the view contents if necessary.

Web application environment definition file (deployment descriptor)

Specify this in the <error-page> tag. For details of the method to specify this, refer to Web Application Environment Definition File (Deployment Descriptor).

This error page is enabled for HTTP error status codes that occur in Web applications.

If this error page is used, the HTTP error status code is not changed. For example, if error-page is set for Exception, the HTTP error status code is 500.

If the response header has already been sent in the Web browser, the information that has already been sent cannot be recovered. For this reason, the HTTP error status code is “Sent”.

Web server environment settings

In the following cases, the error page specified in the Web server environment settings is used because the control does not pass to the Servlet service.

There was an error in the Web application identifier contained in the request URL

This was not a proper request to the Web application

Example

The following examples explain the error pages specified for each location using HTTP error status code.

HTTP Error Status Code 404 (Not Found)

HTTP Error Status Code 500 (Internal Server Error)

HTTP Error Status Code 404 (Not Found)

An error page specified in the Web application environment definition file is used

- There are no contents in the Web application
- The application is Servlet API and the HTTP error status code is set as “404”

A Web server error page is used

- There was an error in the Web application identifier contained in the request URL
- There was a request to the Web application, but there are no contents on the Web server

HTTP Error Status Code 500 (Internal Server Error)

An error page specified in the Web application environment definition file is used

- Exception or Error occurred while the servlet or JSP application was being executed

  Note) This excludes cases in which an error that occurred in the application was caught, or in which JSP error page settings have been made explicitly to allow the application to run normally through error handling.

- The application is Servlet API and the HTTP error status code is set as “500”

A Web server error page is used

This is used in the following cases:

- An abnormality was detected in the Web server connector

  Example: Connection to IJServer is not possible

  A Web server connector timeout occurred

Note

Depending on the Web browser type and settings, the error page that comes with the Web browser might be displayed instead of the intended error page.

Example: Microsoft(R) Internet Explorer 5.x, 6.0

When [Tools] > [Internet Options] > [Advanced Settings] > [View Simple HTTP Error Message] is enabled (as the default value).

Web Application Environment Definition File (Deployment Descriptor)

The Web application environment definition file (deployment descriptor) sets the Web application operating environment.

When multiple Web applications are used, prepare a definition file for each Web application.

Coding Format of the Web Application Environment Definition File (Deployment Descriptor)

The description format of the deployment descriptor is XML and is shown in the following example:

```
--------------------------------------------------------------------------------
--------
  <?xml version="1.0">
  <!DOCTYPE web-app PUBLIC
  "-//Sun Microsystems, Inc.//DTD Web Application 2.3//EN"
  "http://java.sun.com/dtd/web-app_2_3.dtd">
  <web-app>
    <display-name>display_name</display-name>
    <context-param>
      <param-name>name</param-name>
      <param-value>value</param-value>
    </context-param>
    <filter>
      <filter-name>name</filter-name>
      <filter-class>class</filter-class>
      <init-param>
        <param-name>name</param-name>
        <param-value>value</param-value>
      </init-param>
    </filter>
    <filter-mapping>
      <filter-name>value</filter-name>
      <servlet-name>name</servlet-name>
    </filter-mapping>
    <filter-mapping>
      <filter-name>value</filter-name>
      <url-pattern>pattern</url-pattern>
    </filter-mapping>
    <listener>
      <listener-class>class</listener-class>
    </listener>
    <servlet>
      <servlet-name>name</servlet-name>
      <servlet-class>class</servlet-class>  or  <jsp-file>file-name
      </jsp-file>
```

```
    <init-param>
      <param-name>name</param-name>
      <param-value>value</param-value>
    </init-param>
    <load-on-startup>priority</load-on-startup>
    <security-role-ref>
      <role-name>name</role-name>
      <role-link>name</role-link>
    </security-role-ref>
  </servlet>
  <servlet-mapping>
    <servlet-name>name</servlet-name>
    <url-pattern>pattern</url-pattern>
  </servlet-mapping>
  <session-config>
    <session-timeout>time</session-timeout>
  </session-config>
  <mime-mapping>
    <extension>ext</extension>
    <mime-type>mime</mime-type>
  </mime-mapping>
  <welcome-file-list>
    <welcome-file>filename</welcome-file>
  </welcome-file-list>
  <error-page>
    <error-code>code</error-code>  or  <exception-type>type
    </exception-type>
    <location>resource</location>
  </error-page>
  <taglib>
    <taglib-uri>uri</taglib-uri>
    <taglib-location>location</taglib-location>
  </taglib>
  <resource-env-ref>
    <resource-env-ref-name>env-ref-name</resource-env-ref-name>
    <resource-env-ref-type>type</resource-env-ref-type>
  </resource-env-ref>
  <resource-ref>
    <res-ref-name>ref-name</res-ref-name>
    <res-type>type</res-type>
    <res-auth>signon</res-auth>
  </resource-ref>
  <security-constraint>
    <web-resource-collection>
      <web-resource-name>resource-name</web-resource-name>
      <url-pattern>pattern</url-pattern>
      <http-method>method</http-method>
    </web-resource-collection>
    <auth-constraint>
      <role-name>name</role-name>
    </auth-constraint>
    <user-data-constraint>
      <transport-guarantee>guarantee-type</transport-guarantee>
    </user-data-constraint>
```

```
    </security-constraint>
    <login-config>
      <auth-method>method</auth-method>
      <realm-name>name</realm-name>
      <form-login-config>
        <form-login-page>login-page</form-login-page>
        <form-error-page>error-page</form-error-page>
      </form-login-config>
    </login-config>
    <security-role>
      <role-name>name</role-name>
    </security-role>
    <env-entry>
      <env-entry-name>entry-name</env-entry-name>
      <env-entry-value>entry-value</env-entry-value>
      <env-entry-type>entry-type</env-entry-type>
    </env-entry>
    <ejb-ref>
      <ejb-ref-name>ref-name</ejb-ref-name>
      <ejb-ref-type>ref-type</ejb-ref-type>
      <home>ejb-home</home>
      <remote>ejb-remote</remote>
      <ejb-link>name</ejb-link>
    </ejb-ref>
    <ejb-local-ref>
      <description>description</description>
      <ejb-ref-name>name</ejb-ref-name>
      <ejb-ref-type>type</ejb-ref-type>
      <local-home>home</local-home>
      <local>remote</local>
      <ejb-link>link</ejb-link>
    </ejb-local-ref>
  </web-app>
-------------------------------------------------------------------------------
--------
```

Notes on Coding

No definitions other than those described in this manual can be used.

<?xml...> and <!DOCTYPE application-client...> appearing at the beginning provide an XML declaration and document type definition (DTD), and must be stated at the beginning of the deployment descriptor file.

If two-byte Japanese characters are to be used in the deployment descriptor, specify UTF-8 for the character set ("encoding=") in <?xml...>. If a character set other than UTF-8 is specified when two-byte Japanese characters are used, deployment fails. This restriction also applies to comments.

<web-app> and </web-app> are root tags indicating the beginning and end of an XML file. Always specify these tags.

Provide individual tags in the order described above.

When it overlaps and the tag which cannot perform multiple specification is specified, the tag specified at the end becomes effective.

Distinction is made between upper- and lower-case letters (case-sensitive).

Note that, if any definition other than those described in the manual is specified, the Servlet Service may be started without output of an error message.

Windows

The following characters can be used in pathnames:

Alphanumeric characters, '+', '-', '_', '.', ' ', '$', '%', ':', '\', and '~'

Pathnames can be up to 255 bytes long.

Solaris OE Linux

The following characters can be used in pathnames:

Alphanumeric characters, '+', '-', '_', '.', '$', '%', '/', and '~'

Pathnames can be up to 1023 bytes long.

Web Application Environment Definition File Tags

The tags shown in following table can be specified in the Web application environment definition file.

Tags other than web-app can be omitted. Define tags as necessary.

Detailed definitions are possible by setting lower-level tags between the start and end tags of each tag.

Definition Details

Table 6-1 Web Application Environment Definition File Tag Definition Details

Tag	Description	Tag Requirement	Multiple Specification
web-app	Defines the start and end of the Web application environment definition file.	required	Impossible
display	Defines the name of a servlet context.	Optional	Impossible
context-param	Defines the initialization parameters set in the servlet context. It is possible to set and extract information that is common to all servlets of a Web application in the servlet context.	optional	possible
filter	Define a filter class.	Optional	Possible
filter-mapping	Define a target to which the filter class is applied.	Optional	Possible
listener	Define the name of an implementation class that can be used to apply a measure for an event which might occur in a Web application.	Optional	Possible
Servlet	Defines the servlet attributes, such as the initialization parameters and aliases.	Optional	Possible

Tag	Description	Tag Requirement	Multiple Specification
servlet-mapping	Defines servlet mapping associating a URL to a servlet or JSP.	Optional	Possible
session-config	Defines session parameters when session management is used.	Optional	Impossible
mime-mapping	Defines the mime type extracted by the servlet API.	Optional	Possible
welcome-file-list	Defines the welcome file displayed when a file name is not specified in the URL.	Optional	Impossible
error-page	Defines resources corresponding to error codes and Java exception types.	Optional	Possible
taglib	Defines the tag library when embedding original tags with JSP.	Optional	Possible
resource-env-ref	Define the external resource environment referenced by the web applications.	Optional	Possible
resource-ref	Defines the external resource that is referenced by the Web application.	Optional	Possible
security-constraint	Defines access limit to the Web application.	Optional	Possible
login-config	Defines the user authentication method.	Optional	Impossible
security-role	Defines the security role used for access limit	Optional	Possible
env-entry	Defines the environment entry that is referenced by the Web application.	Optional	Possible
ejb-ref	Defines the EJB object that is referenced by the Web application.	Optional	Possible
ejb-local-ref	Defines the EJB object of the local interface that is referenced by the Web application.	Optional	Possible

Web Application Environment Definition File Tag Definitions

This section describes the content of the Web application environment definition file tag settings.

Note that an explanation of tags other than those under discussion has been omitted in the entry example below.

The example of a definition is described by the case of Solaris OE.

In the case of a Windows system, please read a path suitably.

Start and End of Web Application Environment Definition Files

The start and end of Web application environment definition files is defined with the web-app tag.

Entry Format

```
<web-app>
...
</web-app>
```

Entry Example

```
<web-app>
  <context-param>
  ...
  </context-param>
</web-app>
```

The Name of a Servlet Context

The name of a servlet context is defined with the display-name tag.

The specified servlet context name can be obtained by the following method:

javax.servlet.ServletContext.getServletContextName () method

Entry Format

```
<display-name>name</display-name>
```

Entry Example

```
<web-app>
  <display-name>Example Security Constraint</display-name>
</web-app>
```

Servlet Context Initialization Parameters

The servlet context initialization parameters are defined with the context-param tag.

It is possible to set and extract information that is common to all servlets of a Web application in the servlet context. The javax.servlet.ServletContext.getInitParameterNames() method and the javax.servlet.ServletContext.getInitParameter() method are used.

When the same initialization parameter has been defined more than once, only the parameter value specified last is valid.

Entry Format

```
<context-param>
  <param-name>name</param-name>
  <param-value>value</param-value>
</context-param>
```

Tag Details

Table 6-2 Servlet Context Initialization Parameters Tag Details

Tag	Description	Tag Requirement	Multiple Specification
param-name	Defines servlet context initialization parameter names. The parameter name must be entered. If the parameter name is omitted, the value specified by param-value are set to NULL characters.	Required	Impossible
param-value	Defines the value specified in the servlet context initialization parameter. If the parameter value is omitted, NULL characters are set.	Required	Impossible

Entry Example

```
<web-app>
  <context-param>
    <param-name>E-mail</param-name>
    <param-value>taro@fujitsu.co.jp</param-value>
  </context-param>
</web-app>
```

Web application coding example

```
  public doGet(HttpServletRequest req, HttpServletResponse res)
              throws ServletException,IOException {
  String mail = getServletContext().getInitParameter("E-mail");
  }
```

Filter Class

To use the filter function, define a filter class and a target to which the filter class is to be applied. This section describes how to define a filter class.

Define a filter class using the filter tag.

The specified filter initial value can be retrieved with the javax.servlet.FilterConfig.getInitParameter() method.

Entry Format

```
<filter>
  <filter-name>name</filter-name>
  <filter-class>class</filter-class>
  <init-param>
    <param-name>name</param-name>
    <param-value>value</param-value>
  </init-param>
</filter>
```

Tag Details

Table 6-3 Filter Class Tag Details

Tag	Description	Tag Requirement	Multiple Specification
filter-name	Alias of the filter class. This alias is used to find matches of the servlet name or URI pattern defined by filter-mapping. Although it does not become an error when a name is omitted, the definition of this filter tag becomes invalid. When more than one are defined, the tag specified at the end becomes effective.	Required	Impossible
filter-class	Specify the fully qualified name of a mapped filter class. When more than one are defined, the tag specified at the end becomes effective.	Required	Impossible
init-param	Specify a pair consisting of a name and value as initialization parameters. To use multiple parameters, specify them separately using the <init-param> tag.	Optional	Possible
param-name	Set the name of a filter class initialization parameter. This tag is always required when the <init-param> tag is used.	Required	Impossible
param-value	Set the value of a filter class initialization parameter. This tag is always required when the <init-param> tag is used.	Required	Impossible

Entry Example

See the coding example shown in Filter class Application Target.

Filter class Application Target

To use the filter function, define a filter object and a target to which the filter object is to be applied. This section describes how to define a target to which the filter object is to be applied.

Define a filter object application target using the filter-mapping tag.

To a request, the filter-mapping tag indicates the Web container to which the filters are to be applied and the order in which they must be applied.

Multiple filter-mapping tags can be defined. The order in which filters are applied is determined as follows:

<filter-mapping> tags with <url-pattern> elements defined take priority.

If there are multiple <filter-mapping> tags with <url-pattern> elements in web.xml, they are handled in the order in which they are defined.

<filter-mapping> tags with <servlet-name> elements defined take priority.

If there are multiple <filter-mapping> tags with <servlet-name> elements in web.xml, they are handled in the order in which they are defined.

Entry Format

```
<filter-mapping>
  <filter-name>value</filter-name>
  <servlet-name>name</servlet-name>
</filter-mapping>
<filter-mapping>
  <filter-name>value</filter-name>
  <url-pattern>pattern</url-pattern>
</filter-mapping>
```

Tag Details

Table 6-4 Filter Class Application Target Tag Details

Tag	Description	Tag Requirement	Multiple Specification
filter-name	Specify the name of a filter. This name must match the value of a filter-name tag in a filter tag.	Required	Impossible

Tag	Description	Tag Requirement	Multiple Specification
url-pattern	Specify the URI pattern to be mapped with the filter. This tag cannot be used together with the servlet-name tag. The URL is entered as follows: For a specific URL State the name of the URL to be called: Example: /servlet/servlet1 For URLs, each beginning with a specific prefix (path, identifier): Append /* to the end of the prefix. Example: /prefix/* For URLs, each having a specific extension: Enter using the format “*.xxx” . Example: *.do When using "*.xxx" to specify a file with a specific extension, it cannot be specified together with a prefix. Example: /path/*.do cannot be specified. When a file with a specific extension is specified, all files of a Web application become targets.	Required	Impossible
servlet-name	Specify the name of servlet to be mapped with the filter. This tag cannot be used together with the url-pattern tag. Enter the name specified in the servlet-name tag of the servlet tag as the servlet name. If a servlet name that is not specified is entered or the servlet name is omitted, an error occurs and the definition of the filter-mapping tag is disabled.	Required	Impossible

Entry Example

A filter definition for a specific Servlet is shown as follows:

```
<web-app>
  <filter>
    <filter-name>helloWorld</filter-name>           ...alias of a filter
class
    <filter-class>MyHelloWorldFilter</filter-class> ...filter class name
  </filter>
  <filter-mapping>
    <filter-name>helloWorld</filter-name>           ...alias of a filter
class
    <servlet-name>MyHelloWorld</servlet-name>       ...servlet name
  </filter-mapping>
</web-app>
```

A filter definition for a specific URL (the URL of the JSP file in the following example) is shown as follows:

```
<web-app>
  <filter>
    <filter-name>helloWorld</filter-name>            ...alias of a filter
class
    <filter-class>MyHelloWorldFilter</filter-class>  ...filter class name
  </filter>
  <filter-mapping>
    <filter-name>helloWorld</filter-name>            ...alias of a filter
class
    <url-pattern>>/filter.jsp</url-pattern>      ... filter and the URI
pattern to map
  </filter-mapping>
</web-app>
```

A filter definition for a URI pattern is shown as follows. This example uses a wildcard to specify that all resources under "/" be subjected to the filter function.

```
<web-app>
  <filter>
    <filter-name>helloWorld</filter-name>                ...alias of a
filter class
    <filter-class>MyHelloWorldFilter</filter-class>      ...filter class
name
  </filter>
  <filter-mapping>
    <filter-name>helloWorld</filter-name>                ...alias of a
filter class
    <url-pattern>/*</url-pattern>      ... filter and the URI pattern to map
  </filter-mapping>
</web-app>
```

Web application coding example:

```
  import java.io.*;
  import javax.servlet.*;
  import javax.servlet.http.*;

  public class MyHelloWorldFilter implements Filter {
    String company;
    public void doFilter(ServletRequest req, ServletResponse res,
FilterChain chain)
                        throws IOException, ServletException {
      HttpServletResponseWrapper wrapper = new MyResponseWrapper(
                                           (HttpServletResponse)res);
      chain.doFilter(req, wrapper);
      CharArrayWriter writer = new CharArrayWriter();
      writer.write(wrapper.toString().substring(0,
                   wrapper.toString().indexOf("</body>")-1));
      writer.write("<hr>\n");
      writer.write("Copyrights &copy; " + company + "\n");
      writer.write("</body>\n</html>\n");

      PrintWriter out = res.getWriter();
      res.setContentLength(writer.toString().length());
      out.write(writer.toString());
      out.close();
    }
    public void init(FilterConfig config) throws ServletException {
      company = config.getInitParameter("company");
    }

    public void destroy(){}

    class MyResponseWrapper extends HttpServletResponseWrapper {
      private CharArrayWriter output;
      public String toString() {
        return output.toString();
      }
      public MyResponseWrapper(HttpServletResponse response){
        super(response);
        output = new CharArrayWriter();
      }
      public PrintWriter getWriter(){
        return new PrintWriter(output);
      }
    }
  }
```

Listener Class

The listener class is called when a life cycle event occurs. When a life cycle event occurs in a Web application, a defined listener class automatically starts up.

Define a listener class using the listener tag.

If <listener> is specified in the tag library description file (TLD), both listeners are enabled.

Entry Format

```
<listener>
  <listener-class>class</listener-class>
</listener>
```

Tag Details

Table 6-5 Listener Class Tag Details

Tag	Description	Tag Requirement	Multiple Specification
listener-class	Specifies the complete class name for the following events: Start and stop of contexts (javax.servlet.ServletContextListener interface implementation class) Addition, replacement, and deletion of SevletContext attributes (javax.servlet.ServletContextAttributeListener interface implementation class) Creation and deletion of sessions (javax.servlet.http.HttpSessionListener interface implementation class) Addition of attributes to sessions, replacement of attributes, and deletion of attributes from sessions (javax.servlet.http.HttpSessionAttributeListener interface implementation class) If a nonexistent class name is specified, the Web application fails to start.	Required	Impossible

Entry Example

```
<web-app>
  <listener>
    <listener-class>listeners.ContextListener</listener-class>
  </listener>
</web-app>
```

Servlet Attributes

Servlet and JSP attributes are defined with the servlet tag.

It is possible to set aliases, initialization parameters, and startup as servlet attributes. Initialization parameter settings are retrieved using the javax.servlet.ServletConfig.getInitParameterNames() and javax.servlet.ServletConfig.getInitParameter() methods.

When the same startup order (load order) is defined, loading will be performed in the order entered.

Entry Format

When Defining a Servlet

```
<servlet>
  <servlet-name>name</servlet-name>
  <servlet-class>class</servlet-class>
  <init-param>
    <param-name>name</param-name>
    <param-value>value</param-value>
  </init-param>
  <load-on-startup>priority</load-on-startup>
  <security-role-ref>
    <role-name>name</role-name>
    <role-link>name</role-link>
  </security-role-ref>
</servlet>
```

When Defining a JSP File

```
<servlet>
  <servlet-name>name</servlet-name>
  <jsp-file>file-name</jsp-file>
  <init-param>
    <param-name>name</param-name>
    <param-value>value</param-value>
  </init-param>
  <load-on-startup>priority</load-on-startup>
  <security-role-ref>
    <role-name>name</role-name>
    <role-link>name</role-link>
  </security-role-ref>
</servlet>
```

Tag Details

Table 6-6 Servlet Attributes Tag Details

Tag	Description	Tag Requirement	Multiple Specification
servlet-name	Defines the name of the servlet or JSP. Servlet and JSP names are necessary when the servlet tag is used to define servlet attributes, and when the servlet-mapping tag is used to define servlet mapping. In the case of a servlet, the specified name can also be used as an alias. Only the following characters can be used for access from the browser: Alphanumeric characters, '+', '-', '.', '_', '$' When characters are only used for the servlet-mapping tag, XML friendly characters can be used.	Required	Impossible
servlet-class	Defines the complete servlet class name of servlet. Define this tag when defining servlet.	Required (Only when defining servlet)	Impossible
jsp-file	The JSP file name is defined as the partial pathname starting from the root directory of the Web application. Begin the pathname with a forward slash (/). Define this tag when defining the JSP file. Windows When entering the partial pathname, separate each directory with a slash (/), not a backslash (\).	Required (Only when defining JSP file)	Impossible
init-param	Define the servlet initialization parameter.	Optional	Possible
Param-name	Defines the initialization parameter name of the servlet. It is required when defining an init-param tag.	Required	Impossible
Param-value	Defines the value specified in the servlet initialization parameter. It is required when defining an init-param tag.	Required	Impossible

Tag	Description	Tag Requirement	Multiple Specification
load-on-startup	Defines the startup when Servlet Container is started. Order which loads Servlet and JSP. It defines by -2147483648 to 2147483647. Loading proceeds in order from the smallest number to the largest. If 0 is specified, the relevant servlet or JSP is loaded last. If a negative value is specified, the relevant servlet or JSP is not loaded when the Servlet container is activated. If the parameter value is omitted, the relevant servlet or JSP is not loaded when the servlet or JSP is called. When the following values are specified, the servlet or JSP is loaded last. This is also what happens when 0 is specified, as explained above. 1) When a value smaller than -2147483648 is specified, or 2) When a value larger than 2147483647 is specified, or 3) When characters other than numerical values are specified	Optional Default value: Load servlets or JSPs when they are called.	Impossible
security-role-ref	Defines the reference destination of a security role used for servlet code.	Optional	Possible
role-name	Defines the security role name that is used by the Servlet code. It is required when defining a security-role-ref tag. This parameter can be used as a parameter of the javax.servlet.http.HttpServletRequest.isUserInRole() method.	Required (When defining security-role-ref tag)	Impossible
role-link	Defines the name of the security role name specified by <security-role>. It is required when defining a security-role-ref tag.	Required (When defining security-role-ref tag)	Impossible

Entry Example

When Defining a Servlet

```
<web-app>
  <servlet>
    <servlet-name>Hello</servlet-name>
    <servlet-class>com.fujitsu.jservlet.xxx.HelloWorldServlet</servlet-
class>
    <init-param>
      <param-name>message</param-name>
      <param-value>I'm a Hello servlet</param-value>
    </init-param>
    <load-on-startup>10</load-on-startup>
    <security-role-ref>
      <role-name>Administrator</role-name>
      <role-link>Manager</role-link>
    </security-role-ref>
  </servlet>
</web-app>
```

When Defining a JSP File

```
<web-app>
  <servlet>
    <servlet-name>present</servlet-name>
    <jsp-file>/jsp/present.jsp</jsp-file>
    <init-param>
      <param-name>message</param-name>
      <param-value>I'm a Hello JSP</param-value>
    </init-param>
    <load-on-startup>11</load-on-startup>
    <security-role-ref>
      <role-name>Administrator</role-name>
      <role-link>Manager</role-link>
    </security-role-ref>
  </servlet>
</web-app>
```

Servlet Mapping

It is possible to associate a servlet with a different servlet or JSP without displaying the file or servlet of the specified URL.

Servlet mapping of this type is defined with the servlet-mapping tag.

Enter the servlet-mapping tag after the servlet tag defining the servlet or JSP name.

If stated before the servlet tag, the Web application fails to start.

If the same URL is defined for multiple url-pattern tags, the servlet mapping defined last applies.

When the specified URL is valid for multiple servlet mappings, the order of priority is as follows:

When the url-pattern tag is a file or servlet name.

When the url-pattern tag is a prefix (path, identifier). (Longer names have priority.)

When the url-pattern tag is an extension.

Example: If "/index.html" and "*.html" URLs are defined and "/index.html" is accessed, the definition of file name "/index.html" has priority over extension "*.html".

Entry Format

```
<servlet-mapping>
  <servlet-name>name</servlet-name>
  <url-pattern>pattern</url-pattern>
</servlet-mapping>
```

Tag Details

Table 6-7 Servlet Mapping Tag Details

Tag	Description	Tag Requirement	Multiple Specification
servlet-name	Defines the servlet or JSP name to which a request is to be mapped. To a name, the name specified with the servlet-name tag of a servlet tag is described. If a name other than the specified one is stated, the Web application fails to start.	Required	Impossible

Tag	Description	Tag Requirement	Multiple Specification
url-pattern	Defines the URL mapped to the servlet or JSP application. The URL is entered as follows: For a specific URL Enter the name used when calling with a URL Example: /servlet/servlet1 For URLs, each beginning with a specific prefix (path, identifier) Append /* to the end of the prefix. Example: /prefix/* For URLs, each having a specific extension Enter using the format “*.xxx” . Example: *.do When using "*.xxx" to specify a file with a specific extension, it cannot be specified together with a prefix. Example: /path/*.do cannot be specified. When a file with a specific extension is specified, all files of a Web application become targets.	Required	Impossible

Entry Example

A mapping definition for a specific URL is shown as follows:

```
<web-app>
  <servlet>
    <servlet-name>SendMailServlet</servlet-name>
    <servlet-class>SendMailServlet</servlet-class>
  </servlet>
  <servlet-mapping>
    <servlet-name>SendMailServlet</servlet-name>
    <url-pattern>/SendMailServlet</url-pattern>
  </servlet-mapping>
</web-app>
```

A mapping definition for a request where the URL path information has prefix "director" is shown as follows:

```
<web-app>
  <servlet>
    <servlet-name>director</servlet-name>
    <servlet-class>xxx.yyy.DirectorServlet</servlet-class>
  </servlet>
  <servlet-mapping>
    <servlet-name>director</servlet-name>
    <url-pattern>/director/*</url-pattern>
  </servlet-mapping>
</web-app>
```

A mapping definition for a request where the URL ends with ".do" is shown as follows:

```
<web-app>
  <servlet>
    <servlet-name>action</servlet-name>
    <servlet-class>xxx.yyy.ActionServlet</servlet-class>
  </servlet>
  <servlet-mapping>
    <servlet-name>action</servlet-name>
    <url-pattern>*.do</url-pattern>
  </servlet-mapping>
</web-app>
```

Session Parameter

The session parameter is defined with the session-config tag.

The session timeout period can be set as a session parameter.

The session timeout setting is retrieved with the javax.servlet.http.HttpSession.getMaxInactiveInterval() method.

Entry Format

```
<session-config>
  <session-timeout>time</session-timeout>
</session-config>
```

Tag Details

Table 6-8 Session Parameter Tag Details

Tag	Description	Tag Requirement	Multiple Specification
session-timeout	Defines the session timeout period in minutes. Specify a value from 0 to 35791394 in minutes. The default is 30 minutes. If 0, or a negative value is specified, no timeout occurs.	Optional	Impossible

Entry Example

```
<web-app>
  <session-config>
    <session-timeout>30</session-timeout>
  </session-config>
</web-app>
```

Mime Types

The mime type is defined with the mime-mapping tag. The mime type defines the default value of Servlet Container.

By defining this tag, it is possible to set the mime type specific to each Web application. The mime type set with this tag has priority over the default mime type.

If the same mime type is defined more than once, only the mime type specified last is valid. The mime type setting is retrieved with the javax.servlet.ServletContext.getMimeType() method.

Entry Format

```
<mime-mapping>
  <extension>ext</extension>
  <mime-type>mime</mime-type>
</mime-mapping>
```

Tag Details

Table 6-9 Mime Type Tag Details

Tag	Description	Tag Requirement	Multiple Specification
extension	Defines the extension of the file defining the mime type.	Required	Impossible
mime-type	Defines the mime type.	Required	Impossible

Entry Example

```
<web-app>
  <mime-mapping>
    <extension>jpg</extension>
    <mime-type>image/jpeg</mime-type>
  </mime-mapping>
</web-app>
```

Default Mime Type

Table 6-10 Default Mime Type

Extension	Mime Type
abs	audio/x-mpeg
ai	application/postscript
aif aifc aiff	audio/x-aiff
aim	application/x-aim
art	image/x-jg
asf asx	video/x-ms-asf
au	audio/basic
avi	video/x-msvideo
avx	video/x-rad-screenplay
bcpio	application/x-bcpio
bin	application/octet-stream
bmp	image/bmp
body	text/html
cdf	application/x-netcdf
cer	application/x-x509-ca-cert
class	application/java
cpio	application/x-cpio
csh	application/x-csh
css	text/css
dib	image/bmp
doc	application/msword
dtd	text/plain

Extension	Mime Type
dv	video/x-dv
dvi	application/x-dvi
eps	application/postscript
etx	text/x-setext
exe	application/octet-stream
gif	image/gif
gtar	application/x-gtar
gz	application/x-gzip
hdf	application/x-hdf
hqx	application/mac-binhex40
htc	text/x-component
htm html	text/html
hqx	application/mac-binhex40
ief	image/ief
jad	text/vnd.sun.j2me.app-descriptor
jar	application/java-archive
java	text/plain
jnlp	application/x-java-jnlp-file
jpe jpeg jpg	image/jpeg
js	text/javascript
jsf	text/plain
jspf	text/plain
kar	audio/x-midi
latex	application/x-latex
m3u	audio/x-mpegurl
mac	image/x-macpaint
man	application/x-troff-man
me	application/x-troff-me
mid	audio/x-midi
midi	audio/x-midi
mif	application/x-mif

Extension	Mime Type
mov	video/quicktime
movie	video/x-sgi-movie
mp1 mp2 mp3 mpa	audio/x-mpeg
mpe mpeg mpega mpg	video/mpeg
mpv2	video/mpeg2
ms	application/x-wais-source
nc	application/x-netcdf
oda	application/oda
pbm	image/x-portable-bitmap
pct	image/pict
pdf	application/pdf
pgm	image/x-portable-graymap
pic pict	image/pict
pls	audio/x-scpls
png	image/png
pnm	image/x-portable-anymap
pnt	image/x-macpaint
ppm	image/x-portable-pixmap
ps	application/postscript
psd	image/x-photoshop
qt	video/quicktime
qti qtif	image/x-quicktime
ras	image/x-cmu-raster
rgb	image/x-rgb
rm	application/vnd.rn-realmedia
roff	application/x-troff
rtf	application/rtf
rtx	text/richtext

Extension	Mime Type
sh	application/x-sh
shar	application/x-shar
smf	audio/x-midi
snd	audio/basic
src	application/x-wais-source
sv4cpio	application/x-sv4cpio
sv4crc	application/x-sv4crc
svg	image/svg+xml
svgz	image/svg+xml
swf	application/x-shockwave-flash
t	application/x-troff
tar	application/x-tar
tcl	application/x-tcl
tex	application/x-tex
texi texinfo	application/x-texinfo
tif tiff	image/tiff
tr	application/x-troff
tsv	text/tab-separated-values
txt	text/plain
ulw	audio/basic
ustar	application/x-ustar
wav	audio/x-wav
wbmp	image/vnd.wap.wbmp
wml	text/vnd.wap.wml
wmlc	application/vnd.wap.wmlc
wmls	text/vnd.wap.wmlscript
wmlscriptc	application/vnd.wap.wmlscriptc
wrl	x-world/x-vrml
xbm	image/x-xbitmap
xml	text/xml
xpm	image/x-xpixmap

Extension	Mime Type
xsl	text/xml
xwd	image/x-xwindowdump
Z z	application/x-compress
zip	application/zip

Welcome Files

It is possible to define the file to be displayed (welcome file) when no file name is entered in the URL.

The welcome file is valid when the Web application name has been specified in the URL or when a directory name has been specified as a partial pathname starting from the root directory of the Web application.

If the welcome file is omitted, the default file is used. The following files are used as default files:

index.html

index.htm

index.jsp

If a file corresponding to the welcome file (or the default file) is not found, status code 404 (file not found) or the list of directories and files under the corresponding directory is displayed. The file that is displayed depends on the value specified in [Servlet Container Settings] > [List File] on the Interstage Management Console.

Define the welcome file using the welcome-file-list tag. Multiple welcome files can be specified and they are enabled in the order they are included.

Entry Format

```
<welcome-file-list>
  <welcome-file>filename</welcome-file>
</welcome-file-list>
```

Tag Details

Table 6-11 Welcome File Tag Details

Tag	Description	Tag Requirement	Multiple Specification
welcome-file	Defines welcome file.	Required	Possible

Entry Example

```
<web-app>
  <welcome-file-list>
    <welcome-file>index.jsp</welcome-file>
    <welcome-file>index.htm</welcome-file>
  </welcome-file-list>
</web-app>
```

Resources during Error Occurrence

It is possible to define resources (HTML files, servlets) that deal with HTTP errors and Java exceptions.

Resources for error occurrences are defined with the error-page tag.

When resources for the same HTTP error code or Java exception type has been defined more than once, only the last resource definition is valid.

Entry Format

For HTTP Error

```
<error-page>
  <error-code>code</error-code>
  <location>resource</location>
</error-page>
```

For Java Exception

```
<error-page>
  <exception-type>type</exception-type>
  <location>resource</location>
</error-page>
```

Tag Details

Table 6-12 Resources during Error Occurrence Tag Details

Tag	Description	Tag Requirement	Multiple Specification
error-code	Defines the HTTP error code. Define either the error-code tag or the exception-type tag.	Required	Impossible
exception-type	Defines the complete class name of the Java exception type. Define either the error-code tag or the exception-type tag.	Required	Impossible

Tag	Description	Tag Requirement	Multiple Specification
location	Defines resources (HTML documents, servlets, etc.) that respond to errors. Specify the partial pathname starting from the root directory of the Web application. Add / to the beginning of the pathname. Omitting the resource results in an error. Windows When entering directories in the partial pathname, separate each directory name with a slash (/), not a backslash (\). Note) An error page with a built-in web browser may be displayed if a web browser is set.	Required	Impossible

Either an error-code tag or an exception-type tag is defined.

When both are not specified, this error-page tag becomes invalid.

Entry Example

For HTTP Error

```
<web-app>
  <error-page>
    <error-code>500</error-code>
    <location>/error/http/code500.html</location>
  </error-page>
</web-app>
```

For Java Exception

```
<web-app>
  <error-page>
    <exception-type>java.lang.IllegalStateException</exception-type>
    <location>/error/exception/IllegalState.html</location>
  </error-page>
</web-app>
```

JSP Tag Libraries

JSP tag libraries are defined with the taglib tag.

Entry Format

```
<taglib>
  <taglib-uri>uri</taglib-uri>
  <taglib-location>location</taglib-location>
</taglib>
```

Tag Details

Table 6-13 JSP Tag Library Tag Details

Tag	Description	Tag Requirement	Multiple Specification
taglib-uri	Defines the URI of the tag library of the JSP used by the Web application. Specify the URL to be defined in uri, which is specified by <taglib> in the JSP file.	Required	Impossible
taglib-location	Defines the name of the Tag Library Description file (TLD) of the tag library. Specify the partial pathname starting from the root directory of the Web application. Add / to the beginning of the pathname. If a nonexistent path is stated, the Web application fails to start. Windows When entering directories in the partial pathname, separate each directory name with a slash (/), not a backslash (\).	Required	Impossible

Entry Example

```
<web-app>
  <taglib>
    <taglib-uri>http://java.apache.org/tomcat/examples-taglib</taglib-uri>
    <taglib-location>/WEB-INF/jsp/example-taglib.tld</taglib-location>
  </taglib>
</web-app>
```

Entry Example of JSP File

```
<html>
<body>
<%@ taglib uri="http://java.apache.org/tomcat/examples-taglib"
prefix="eg" %>
Radio stations that rock:
```

```
<ul>
<eg:foo att1="98.5" att2="92.3" att3="107.7">
<li>
<%= member %>
</li>
</eg:foo>
</ul>
    .
    .
```

Note

When <tag-class> tag in the tag library descriptor file is changed, the JSP file using the tag library corresponding to the changed tag library descriptor must be recompiled.

The JSP is recompiled when the java source file and class file corresponding to the JSP file do not exist in the work directory under the IJServer directory.

Therefore, if the java source file and the class file that correspond to the JSP file are deleted, JSP recompilation is executed.

For example, when the JSP file path from the root directory of the web application is "/jsp/HelloJSP.jsp", the source file and the class file are created as follows:

Source file name: jsp\HelloJSP_jsp.java

Class file name: jsp\HelloJSP_jsp.class

External Resource Environment Reference

Define the external resource environment referenced by the web application using the resource-env-ref tag.

Entry Format

```
<resource-env-ref>
  <resource-env-ref-name>env-ref-name</resource-env-ref-name>
  <resource-env-ref-type>type</resource-env-ref-type>
</resource-env-ref>
```

Tag Details

Table 6-14 External Resource Environment Reference Tag Details

Tag	Description	Tag Requirement	Multiple Specification
resource-env-ref-name	Any reference name can be specified so that the external resource environment can be identified by the web application. Define the name relative to the java:comp/env context. Example: jms/xxxxx	Required	Impossible
resource-env-ref-type	Define the type of data source of the external resource environment reference. Specify the JavaClass type as its type.The following resources are supported as the data source type (JavaClass type): JMS javax.jms.Topic javax.jms.Queue	Required	Impossible

Entry Example

```
<web-app>
  <resource-env-ref>
    <resource-env-ref-name>jms/sTopic</resource-env-ref-name>
    <resource-env-ref-type>javax.jms.Topic</resource-env-ref-type>
  </resource-env-ref>
</web-app>
```

Defining References to External Resources

External resources that are referenced by the Web application are defined with the resource-ref tag.

Entry Format

```
<resource-ref>
  <res-ref-name>ref-name</res-ref-name>
  <res-type>type</res-type>
  <res-auth>signon</res-auth>
</resource-ref>
```

Tag Details

Table 6-15 Defining References to External Resource Tag Details

Tag	Description	Tag Requirement	Multiple Specification
res-ref-name	The reference name can optionally be specified to let the Web application identify the external resource. Define the name relative to the java:comp/environment context. Example: For JDBC jdbc/xxxxx For JMS jms/xxxxx For JavaMail mail/xxxxx For URL url/xxxxx	Required	Impossible
res-type	Defines the data source type of the external resource. Specify the JavaClass type. The data source type (JavaClass type) supports the following resources: - JDBC javax.sql.DataSource - JavaMail javax.mail.session - JMS javax.jms.TopicConnectionFactory javax.jms.QueueConnectionFactory - URL java.net.URL	Required	Impossible

Tag	Description	Tag Requirement	Multiple Specification
res-auth	Defines whether the resource connection information is specified by the application component code in the program or Servlet Container. See "Resource-connectable User Control Function" in Chapter 4 for more information. Define the position where the connection information for the resource is set up. Application or Container can be specified. • Application The connection information is set by the application component code (Web application side). • Container The connection information is set by the Container. The container uses the connection information set by resource definitions.	Required	Impossible

Entry Example

```
<web-app>
  <resource-ref>
    <res-ref-name>jdbc/MyDataBase</res-ref-name>
    <res-type>javax.sql.DataSource</res-type>
    <res-auth>Container</res-auth>
  </resource-ref>
</web-app>
```

Access Limit

The Access limit is defined with the security-constraint tag.

Entry Format

```
<security-constraint>
  <web-resource-collection>
    <web-resource-name>resource-name</web-resource-name>
    <url-pattern>pattern</url-pattern>
    <http-method>method</http-method>
  </web-resource-collection>
  <auth-constraint>
    <role-name>name</role-name>
  </auth-constraint>
  <user-data-constraint>
    <transport-guarantee>guarantee-type</transport-guarantee>
```

```
  </user-data-constraint>
</security-constraint>
```

Tag Details

Table 6-16 Access Limit Tag Details

Tag	Description	Tag Requirement	Multiple Specification
web-resource-collection	Defines the Web resource collection.	Required	Possible
web-resource-name	Defines the Web resource collection name. It is required when defining web-resource-collection tag.	Required	Impossible
url-pattern	Defines the URL pattern. Define it using the target path from root directory of Web application. Add "/" to the top. Omitting the url-pattern tag or the URL pattern will disable the access constraint specified in the <security-constraint> tag including this tag.	Optional	Possible
http-method	Defines an HTTP method (e.g., GET and POST). Access is limited only to the defined method. If the method is omitted, access limit is applied to all methods.	Optional	Possible
auth-constraint	Defines the security role that allows access to the Web resource collection. The users who have the specified role are allowed to access the resource collection. When the security role is omitted, all users are allowed to access the resource collection.	Optional	Impossible
role-name	Defines the security role name. When the security role name is specified, user authentication is performed since user identification is required. The role specified here must be defined in <role-link> of <security-role-ref> of the <servlet> tag. When the role-name tag is omitted, all users are allowed to access the resource. Please be sure to specify a role-name. When a role- name is omitted, no user can access.	Optional	Possible

Tag	Description	Tag Requirement	Multiple Specification
user-data-constraint	Defines the data security attribute Define the method of protecting data that is communicated between client and container.	Optional	Impossible
transport-guarantee	Defines the transfer method between client and server. It is required when defining user-data-constraint tag NONE Indicates that the application does not require transfer guarantee. - INTEGRAL - CONFIDENTIAL	Required	Impossible

Entry Example

```
<web-app>
  <security-constraint>
    <web-resource-collection>
      <web-resource-name>Hello</web-resource-name>
      <url-pattern>/Hello.jsp</url-pattern>
      <http-method>GET</http-method>
    </web-resource-collection>
    <auth-constraint>
      <role-name>Administrator</role-name>
    </auth-constraint>
    <user-data-constraint>
      <transport-guarantee>NONE</transport-guarantee>
    </user-data-constraint>
  </security-constraint>
</web-app>
```

User Authentication

The user authentication method is defined with the login-config tag.

Entry Format

```
<login-config>
  <auth-method>method</auth-method>
  <realm-name>name</realm-name>
  <form-login-config>
    <form-login-page>login-page</form-login-page>
    <form-error-page>error-page</form-error-page>
  </form-login-config>
</login-config>
```

Tag Details

Table 6-17 User Authentication Tag Details

Tag	Description	Tag Requirement	Multiple Specification
auth-method	Defines the authentication method. Specify one of the following: • BASIC HTTP BASIC authentication • FORM Form-based authentication If the auth-method tag or the authentication method is omitted, "BASIC" becomes the optional value.	Optional	Impossible
realm-name	Defines the area name used for HTTP Basic authentication. The area name is displayed on the screen for user authentication (dialog box). When the HTTP Basic authentication is not used, the specification is ignored.	Optional	Impossible
form-login-config	Defines the start and end of the form-based authentication definitions. Specify the login page and error page used for form-based authentication. When form-based authentication is not used, the specification is ignored.	Optional	Impossible

Tag	Description	Tag Requirement	Multiple Specification
form-login-page	Defines the login page used for form-based authentication. It is required when the form-login-config tag is defined. The specified login page is displayed in the Web browser at form-based authentication. **Note** The login page must use the following interface to pass the user name and password to the Servlet Container: - Application name: j_security_check - Parameter name -> user name: j_username - Parameter name -> password: j_password **Example** : <FROM ACTION="j_security_check" METHOD="POST"> UserName: <INPUT TYPE="text" NAME="j_username"> Password: <INPUT TYPE="password NAME="j_password"> </FORM> : When form-based authentication is specified with an auth-method tag, be sure to specify this tag. Be sure to specify a login page. If omitted, login page is not displayed.	Required	Impossible
form-error-page	Defines the location of the error page that is displayed when form-based authentication fails. The specified error page is displayed on the Web browser at failure of form-based authentication.	Required (If the form-login-config tag is defined)	Impossible

Entry Example

For HTTP BASIC authentication

```
<web-app>
  <login-config>
    <auth-method>BASIC</auth-method>
    <realm-name>Welcome Page</realm-name>
  </login-config>
</web-app>
```

For Form-based authentication

```
<web-app>
  <login-config>
    <auth-method>FORM</auth-method>
    <form-login-config>
      <form-login-page>/login.jsp</form-login-page>
      <form-error-page>/error.jsp</form-error-page>
    </form-login-config>
  </login-config>
</web-app>
```

Attestation Continuation Processing

The operation for continuing form-based authentication varies, depending on the setting of "Save session information on client's web browser" specified for the Web application.

Use the Interstage Management Console to specify the setting of the Web application.

If "Save session information on client's web browser" is enabled:

Input processing of a user name/password is required only the first time a client is started. Attestation is continued even if it ends a client.

The continuation time of attestation is decided by the following.

- The timeout of session

  Attestation is continued until session carries out a timeout.

  The session-config tag of a Web application environmental definition file(web.xml) or

  the setMaxInactiveInterval (int interval) method of a HttpSession class can define the timeout time of session.

  Refer to "Session Parameter" for the details of a session-config tag.

- Until application cancels session

  Attestation is continued until it cancels session by the processing in application using the invalidate() method of a HttpSession class.

Note

Attestation continuation processing in form base attestation is mounted using a Cookie. If the client does not support cookies or has them disabled, authentication is not continued even if "Save session information on client's web browser" is enabled.

If "Save session information on client's web browser" is disabled

Whenever input processing of a user name/password starts a client, it is required.

Attestation becomes invalid after ending a client.

Security Role

The security role is defined with the security-role tag.

Entry Format

```
<security-role>
  <role-name>name</role-name>
</security-role>
```

Tag Details

Table 6-18 Security Role Tag Details

Tag	Description	Tag Requirement	Multiple Specification
role-name	Defines the security role name. For the role name, specify the security role name specified in the operation setup for the security function.	Required	Impossible

Entry Example

```
<web-app>
  <security-role>
    <role-name>Administrator</role-name>
  </security-role>
</web-app>
```

Application Environment Entry

The application environment entry is defined with the env-entry tag.

Entry Format

```
<env-entry>
  <env-entry-name>entry-name</env-entry-name>
  <env-entry-value>entry-value</env-entry-value>
  <env-entry-type>entry-type</env-entry-type>
</env-entry>
```

Tag Details

Table 6-19 Application Environment Entry Tag Details

<table>
<tr><th>Tag</th><th>Description</th><th>Tag Requirement</th><th>Multiple Specification</th></tr>
<tr><td>env-entry-name</td><td>Defines the entry name of the environment entry. Define the name relative to the java:comp/env context.</td><td>Required</td><td>Impossible</td></tr>
<tr><td>env-entry-value</td><td>Defines the entry value of the environment entry. The entry value is handled as an object specified in <env-entry-type>.

An error occurs when an entry value differs from the Java type specified with the env-entry-type tag.
If the tag is omitted, the following operation is performed:
<table>
<tr><td>env-entry-type</td><td>Entry Value</td></tr>
<tr><td>java.lang.Boolean</td><td>Boolean.FALSE</td></tr>
<tr><td>java.lang.Byte</td><td>Byte object with value 0</td></tr>
<tr><td>java.lang.Character</td><td>Character object with value 0</td></tr>
<tr><td>java.lang.String</td><td>None
*1
javax.naming.NameNotFoundException is issued.</td></tr>
<tr><td>java.lang.Short</td><td>Short object with value 0</td></tr>
<tr><td>java.lang.Integer</td><td>Integer object with value 0</td></tr>
<tr><td>java.lang.Long</td><td>Long object with value 0</td></tr>
<tr><td>java.lang.Float</td><td>Float object with value 0</td></tr>
<tr><td>java.lang.Double</td><td>Double object with value 0</td></tr>
</table></td><td>Optional</td><td>Impossible</td></tr>
</table>

Tag	Description	Tag Requirement	Multiple Specification
env-entry-type	Defines the Java type of the environment entry that is handled by the application code. The following Java types can be specified: Java.lang.Boolean java.lang.Byte java.lang.Character Java.lang.String java.lang.Short Java.lang.Integer java.lang.Long java.lang.Float Java.lang.Double Java.lang.Float	Required	Impossible

Entry Example

```
<web-app>
  <env-entry>
    <env-entry-name>company</env-entry-name>
    <env-entry-value>Fujitsu</env-entry-value>
    <env-entry-type>java.lang.String</env-entry-type>
  </env-entry>
</web-app>
```

EJB Object Reference

The EJB object reference is defined with the ejb-ref tag.

Entry Format

```
<ejb-ref>
  <ejb-ref-name>ref-name</ejb-ref-name>
  <ejb-ref-type>ref-type</ejb-ref-type>
  <home>ejb-home</home>
  <remote>ejb-remote</remote>
  <ejb-link>name</ejb-link>
</ejb-ref>
```

Tag Details

Table 6-20 EJB Object Reference Tag Details

Tag	Description	Tag Requirement	Multiple Specification
ejb-ref-name	Defines the reference name of the EJB object. Define the name relative to the java:comp/env context. Example: ejb/xxxxx	Required	Impossible
ejb-ref-type	Defines the EJB object type. The following types can be specified: • Entity Entity Bean • Session Session Bean	Required	Impossible
home	Defines the full name of the EJB home interface.	Required	Impossible
remote	Defines the full name of the EJB remote interface.	Required	Impossible
ejb-link	Defines the EJB name (ejb-name) in J2EE application PKG to be linked.	Optional	Impossible

Entry Example

```
<web-app>
  <ejb-ref>
    <ejb-ref-name>ejb/EjbTest</ejb-ref-name>
    <ejb-ref-type>Session</ejb-ref-type>
    <home>com.fujitsu.interstage.ejb.EjbTestHome</home>
    <remote>com.fujitsu.interstage.ejb.EjbTest</remote>
    <ejb-link>EjbTest</ejb-link>
  </ejb-ref>
</web-app>
```

EJB object reference of Local interface

Use the ejb-local-ref tag to define the EJB object reference of the Local interface.

Entry Format

```
<ejb-local-ref>
  <ejb-ref-name>ref-name</ejb-ref-name>
  <ejb-ref-type>ref-type</ejb-ref-type>
  <local-home>ejb-local-home</local-home>
  <local>ejb-local</local>
  <ejb-link>name</ejb-link>
</ejb-local-ref>
```

Tag Details

Table 6-21 EJB Object Reference Tag Details

Tag	Description	Tag Requirement	Multiple Specification
ejb-ref-name	Defines the reference name of the EJB object. Define the name relative to the java:comp/env context. Example: ejb/xxxxx	Required	Impossible
ejb-ref-type	Defines the EJB object type. The following types can be specified: • Entity Entity Bean • Session Session Bean	Required	Impossible
local-home home	Defines the full name of the EJB local home interface.	Required	Impossible
local	Defines the full name of the EJB local interface.	Required	Impossible
ejb-link	Defines the EJB name (ejb-name) in J2EE application PKG to be linked.	Optional	Impossible

Entry Example

```
<web-app>
  <ejb-ref>
    <ejb-ref-name>ejb/EjbTest</ejb-ref-name>
    <ejb-ref-type>Session</ejb-ref-type>
    <home>com.fujitsu.interstage.ejb.EjbTestHome</home>
    <remote>com.fujitsu.interstage.ejb.EjbTest</remote>
    <ejb-link>EjbTest</ejb-link>
  </ejb-ref>
</web-app>
```

Chapter 7

How to Call Web Applications

This chapter describes how to call Web applications.

The definition examples are for the Solaris OE.

If you are using a Windows system, change the path to one that is suitable for reading in a Windows system.

Calling Servlets

Servlets are called by Web browser URLs and by specifying URLs in links in HTML text.

A servlet can be called by specifying the following for each IJServer:

- Call that requires mapping

  The servlet cannot operate without mapping.

- Call that does not require mapping

  The servlet can operate without mapping.

For security reasons, it is generally recommended to use the calling method that requires mapping.

By default the calling method that does not require mapping is disabled. If it is needed, from the Interstage Management Console, select [System] > [WorkUnit] > [IJServer name] > [Environment Settings] > [Servlet Container Settings] and change the setting of [Servlet operates even without mapping].

Call that Requires Mapping

Servlet URL

The Servlet URL corresponds to the URL pattern defined in "Servlet mapping definition" (servlet-mapping url-pattern tag) in "Web application environment definition file (deployment descriptor)."

Servlet Stored Directory

Servlets are stored in one of the following directories:

- Web application root directory/WEB-INF/classes
- Web application root directory/WEB-INF/lib JAR file.

When a Servlet is Created as a Package

The package name is added to the servlet name and called. Packages can be created by specifying the package statement in the source code of the servlet.

If the package name is org.xxx.zzzz and the servlet name is HelloWorldServlet, the call name is as follows:

```
org.xxx.zzzz.HelloWorldServlet
```

Calling by Specification in a URL

```
http://Server-host-name:Port-No./Web-application-name/servlet-URL
```

Note

The port number can be omitted. In this case, 80 is used as the port number.

Calling from within an HTML Document

```
<A HREF="/Web-application-name/servlet-URL">Click Here</A>
```

It is also possible to create an input field in HTML text (HTML FORM tag) and transfer information. Servlets are then called in the following way. METHOD can also be obtained by specifying GET.

```
<FORM ACTION="/Web-application-name/servlet-URL" METHOD=POST>
```

To Pass Information to a Servlet

To pass parameters to a servlet, specify them in the following format after the servlet URL.

```
Servlet URL?Parameter name 1 = Value 1 & Parameter name 2 = Value 2 & ...
```

Similarly to CGI, PATH_INFO can be used to transfer path information to the servlet. Specify the path information after the servlet name, beginning with /.

When completing this step, make sure that you add "/*" to the end of the url-pattern tag of the Servlet mapping definition.

```
Servlet URL/Path information?Parameter name 1 = Value 1 & Parameter name 2 = Value
2 & ...
```

Notes

- When only a status code and message are displayed in a Web browser (and not the results of running a servlet), there may be an error in the environment settings for the Web server, or in the way the servlet is called.
- If servlets with the same name are located in WEB-INF/classes and a WEB-INF/lib JAR file, the servlet in WEB-INF/classes will be called.
- Windows

  In a Windows(R) system, if the name of a specified servlet contains an incorrect case letter, the Java exception java.lang.NoClassDefFoundError occurs.

  In this case, "500 Internal Server Error" is displayed by the Web browser.

  It is recommended to create error pages to notify the user of the incorrect case letters and "404 Not Found," and define these pages in the "Resources when an error occurs" (error-page tag) of "Web application environment definition file (deployment descriptor)."

Call That Does Not Require Mapping

Servlet name

Servlets are called by specifying the servlet name. The servlet name refers to the file name excluding the extension .class. The case (upper or lower) of the characters used in the servlet name is significant.

Servlet stored directory

Servlets are stored in one of the following directories:

- Web application root directory/WEB-INF/classes
- Web application root directory/WEB-INF/lib JAR file

When a servlet is created as a package

The package name is added to the servlet name and called. Packages can be created by specifying the package statement in the source code of the servlet.

If the package name is org.xxx.zzzz and the servlet name is HelloWorldServlet, the call name is as follows:

```
org.xxx.zzzz.HelloWorldServlet
```

Calling by Specification in a URL

```
http://Server-host-name:Port-No./Web-application-name/servlet/servlet-name
```

Note

The port number can be omitted. In this case, 80 is used as the port number.

Calling from within an HTML Document

```
<A HREF="/Web-application-name/servlet/servlet-name">Click Here</A>
```

It is also possible to create an input field in HTML text (HTML FORM tag) and transfer information. Servlets are then called in the following way. METHOD can also be obtained by specifying GET.

```
<FORM ACTION="/Web-application-name/servlet/servlet-name" METHOD=POST>
```

To Pass Information to a Servlet

To pass parameters to a servlet, specify them in the following format after the servlet name.

```
Servlet name?Parameter name 1 = Value 1 & Parameter name 2 = Value 2 & ...
```

Similarly to CGI, PATH_INFO can be used to transfer path information to the servlet. Specify the path information after the servlet name, beginning with /.

```
Servlet name/Path information?Parameter name 1 = Value 1 & Parameter name 2 =
Value 2 & ...
```

Specifying an Alias

The servlet name can be specified as an alias. The alias is defined with the servlet tag in the Web application environment definition file. Refer to Servlet Attributes in Chapter 6 for an explanation of alias settings. Below is an example using the servlet name and an example using an alias.

Example using the servlet name

```
http://hostname/webapl1/servlet/HelloWorldServlet
```

Example using an alias (“HelloWorldServlet” defined by the alias “Hello”.)

```
http://hostname/webapl1/servlet/Hello
```

Notes

- When only a status code and message are displayed in a Web browser (and not the results of running a servlet), there may be an error in the environment settings for the Web server, or in the way the servlet is called.
- If servlets with the same name are located in WEB-INF/classes and a WEB-INF/lib JAR file, the servlet in WEB-INF/classes will be called.
- Windows

  In a Windows(R) system, if the name of a specified servlet contains an incorrect case letter, the Java exception java.lang.NoClassDefFoundError occurs.

  In this case, "500 Internal Server Error" is displayed by the Web browser.

  It is recommended to create error pages to notify the user of the incorrect case letters and "404 Not Found," and define these pages in the "Resources when an error occurs" (error-page tag) of "Web application environment definition file (deployment descriptor)."

Calling JSPs

JSPs are called by Web browser URLs and by specifying URLs in links in HTML text.

The Partial Pathname of the JSP

Servlets are called by specifying the JSP file name. The partial pathname starting from the Web application root directory is specified in the URL. This is referred to from now on as “the partial pathname of the JSP”.

Below are examples of full and partial pathnames of a JSP.

Full pathname:

```
Web application root directory/jsp/Hello/HelloJSP.jsp
```

Partial pathname:

```
jsp/Hello/HelloJSP.jsp
```

The following is an explanation of the methods used to call a JSP.

Calling by Specification in a URL

```
http://Server-host-name:Port-No./Web-application-name/Partial-pathname-JSP
```

Note

The port number can be omitted. In this case, 80 is used as the port number.

Sample

```
http://hostname/webapl1/jsp/Hello/HellpJSP.jsp
```

Calling from within an HTML Document

```
<A HREF="/Web-application-name/Partial-pathname-JSP">Click Here</A>
```

Note

When only a status code and message are displayed in a Web browser (and not the results of running a JSP), there may be an error in the environment settings for the Web server, or in the way the JSP is called.

Calling HTML, Image and Other Files

HTML files, image files, and other files are called by specifying a URL in a Web browser or a link in an HTML document.

The Partial Pathname of the file

The URL is specified as the partial pathname starting from the root directory of the Web application. This is referred to from now on as “the partial pathname of the file”.

Below are examples of full and partial pathnames of a file.

Full pathname:

```
Web application root directory/apl/Hello/index.htm
```

Partial pathname:

```
apl/Hello/index.htm
```

Files are called by the following methods.

Calling by Specification in a URL

```
http://Server-host-name:Port-No./Web-application-name/Partial-pathname-file
```

Note

The port number can be omitted. In this case, 80 is used as the port number.

Sample

```
http://hostname/webapl1/apl/Hello/index.htm
```

Calling from within an HTML Document

```
<A HREF="/Web-application-name/Partial-pathname-file">Click Here</A>
```

Part III

EJB Edition

Chapter 8

Basic Functions of the EJB Service

This chapter explains the following topics:

- Session Bean Time Monitoring
- Performance Option
- Maximum Time Monitoring Function for Application Processing
- Setting Values for Individual Time Monitoring Functions
- Waiting Time Monitoring Function for Server Return
- Idle-time monitoring function of STATEFUL Session Bean
- Setting Values for Individual Time Monitoring Functions
- Timer deletion of EJB object
- Notes in EJB Service

Session Bean Time Monitoring

If no business method is executed for an EJB object of the STATEFUL Session Bean after a given period, the EJB Service lets the container delete the EJB object corresponding to the relevant instance.

This function is called the "STATEFUL Session Bean idle-time monitoring function".

Refer to "STATEFUL Session Bean no-communication monitoring function" for details of the function.

Note

The no-communication monitoring function is not required for the STATELESS Session Bean because it reuses EJB objects.

Managing Entity Bean Instances

Setting the Number of Instances

In conformance with the EJB specification, the EJB Service performs pool management of the area used for Entity Bean instances in virtual memory. The user can set the number of instances to be pool-managed. An instance is created at timing set in instance creation mode and held until it is stopped and maintained until the application terminates. Performing data manipulations exceeding the set number of instances will not result in an application operation error, but it will affect processing performance slightly as the database will be accessed.

Instance Management Mode

To improve processing and memory performance, the EJB Service provides an instance management mode that can be made to suit the user's needs.

The types of instance management and their uses are shown in the following table.

Table 8-1 Instance Management Uses

Instance management mode	Use
ReadWrite (Default)	Data access within the same transaction is improved by caching the instances for each transaction. This mode is effective when performing searches and database updates online.
ReadOnly	High-speed searches are made possible by caching instances that extend over more than one transaction. This mode is effective in cases such as conducting an online search of master information that will not be updated.
Sequential	Memory performance is improved with respect to processes involving data being extracted and manipulated sequentially. This mode is effective when performing batch processing of large quantities of data.

Instance creation Mode

The timing of the creation of the specified number of instances can be selected for the Entity Bean.

The Entity Bean instance creation timing is explained below:

Instance creation option	Timing
At Start-Up	The number of Entity Bean instances created at startup.
At First Access	The number of Entity Bean instances created as a result of Entity Bean activation is equal to the number specified for initial activation.
As Required	If a necessary Entity Bean instance is not present in the accessed management pool after completing Entity Bean activation, the instance needs to be created. The created instance is stored in the pool as a result of deactivation (passivation). This mode is set as the default value.

Note

The maximum number of instances that can be created is specified in "The instances of Entity Bean".

To set the instance creation option, from the Interstage Management Console, select [WorkUnit] > [IJServer name] > [EJB application] > [Application environment definition] > [Interstage extended information].

The table below lists the standards of instance creation mode selection for the Entity Bean.

Table 8-2 Instance creation

Instance creation option	When Selected
At Start-Up	1) Select this mode when processing performance is to be improved immediately after starting Entity Bean operation.
At First Access	1) Select this mode when the Entity Bean activation performance is to be improved.
As Required	1) Select this mode when the Entity Bean activation performance is to be improved. 2) Select this mode when the Entity Bean is not often accessed.

Note

The number of instances, instance management mode and instance creation mode can be set for each Entity Bean.

Entity Bean Optimization

When creating an Entity Bean of BMP using Apworks, a high-performance Entity Bean can be created by selecting "Optimize Entity Bean" in the generation wizard of the Enterprise Bean. Optimization of the Entity Bean can be used if the following conditions are met.

- The transaction attribute of the Entity Bean is either "Mandatory" or "Required".
- A transaction is not completed during multiple searching
- A connection is not cut off during multiple searching
- The Entity Bean is deployed in an EJB container by using a Light EJB container that uses the local call.

Note that, if the optimization function is used under any condition that does not match the above conditions, a malfunction may occur in Entity Bean processing.

Note

An application that uses the optimization function cannot use the distributed transaction function. If it uses the distributed transaction function, the SQLException (ORA-01002:invalid the fetch order) message will be returned when the following actions are executed.

- For the return value, executing a finder method of Enumeration or Collection
- For the return value Enumeration, executing nextElement method or for the return value Iterator of Collection, executing next method.
- Executing nextElement or next for the number of the records that are fetched once in Oracle (default value is 10).

EJB QL

Note

Note the following points when the DBMS that is used is Symfoware.

- The SQRT function and MOD function cannot be used
- The third argument of the LOCATE function cannot be specified
- An input parameter cannot be specified in the LENGTH function
- The input parameter cannot be specified in the NULL comparison expression.
- An input parameter cannot be specified in the first argument of the SUBSTRING function.

What is a Message-driven Bean?

JMS Destination and JMS ConnectionFactory definitions

To run a Message-driven Bean, define "Destination name" and "JMS Connection Factory" on [Message-driven Bean extended information]. To do so, from the Interstage Management Console, select [WorkUnit] > [IJServer name] > [EJB application] > [Application environment definition] > [Message-driven Bean extended information].

The default value of each definition is as follows:

Definition name		Default value
JMS Connection Factory name	Topic	TopicCF001
	Queue	QueueCF001
Destination name		EJB application name

Durable Subscription Function

To use this function, do as follows: From the Interstage Management Console, select [WorkUnit] > [IJServer name] > [EJB application] > [Application environment definition] > [Message-driven Bean extended information], and set "NonDurable" or "Durable" on [Subscriber persistence]. Refer to Help for the Interstage Management Console for details of the setting procedure.

Register and Delete Durable Subscriber Definition

If "Durable" is specified on [Subscriber persistence] that appears by selecting [WorkUnit] > [IJServer name] > [EJB application] > [Application environment definition] > [Message-driven Bean extended information] from the Interstage Management Console, specify "Subscriber identifier." IJServer registers the Durable Subscriber with the name specified for "Subscriber identifier" when the Message-driven Bean is started for the first time.

When Message-driven Bean is activated again, the maintained message is delivered. When durable Subscriber becomes unnecessary, it is necessary to delete it by the following commands. Refer to JMS Operation Commands in the Reference Manual (Command Edition) for details of the command.

Example

Deleting the Durable Subscriber name "dsub" and a client identifier "client1":

```
jmsrmds -n dsub -i client1
```

Message Backup Function in Abnormal Circumstances

When the transaction management type is Container, and the transaction attribute is Required, if a System exception such as RuntimeException or Error occurs in a Message-driven Bean, the container does a transaction rollback.

In this case, the message that the rollback is done is delivered to the Message-driven Bean again, and there is a possibility that it loops.

The message backup function can be used to prevent this.

If a system exception such as RuntimeException and error continuously occurs exceeding the retry count, the container sends a message to a destination for back up.

In case neither the JMS ConnectionFactory name nor the Destination name is specified or they are wrong etc, then the message will be serialized. The process is stopped if the serialization fails.

Use the Interstage Management Console to set "Retry count," "JMS Connection Factory name," and "Destination name" on [Error message save definition].

Refer to Help for the Interstage Management Console for details of the setting procedure.

The filename and storage directory name of the serialized message are shown as follows.

- The filename of serialize file

```
MSG_[ EJB application]_[ processID]_[ threadID]_[ number].ser
```

- The storage directory name of serialize file

  Windows

```
C:/Interstage/EJB/var
```

  Solaris OE Linux

```
/opt/FJSVejb/var
```

The processing image when this function is used is shown as follows.

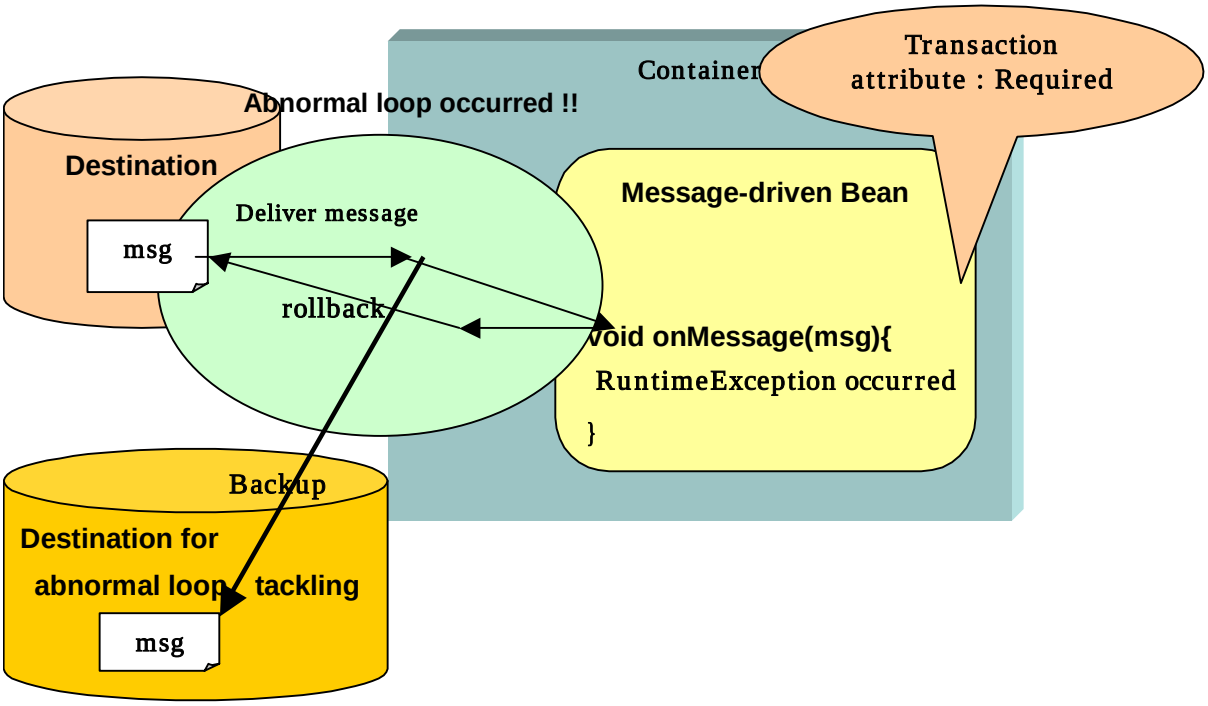


Figure 8-1 Message Backup Function

How to Restore the Serialized Message

Example

The description example to restore the serialized JMS message is shown.

```
FileInputStream fis = new FileInputStream("serialize file name ");
try {
  ObjectInputStream ois = new ObjectInputStream(fis);
  Message msg = (Message)ois.readObject(); // The message that this msg is
backuped
} finally {
  fis.close();
}
```

Notes

In the following cases, there is a possibility that the target message is not the message for backup in abnormal circumstances, so do not use it.

- When other reception applications exist by using the Point-to-Point model in Destination of Message-driven Bean.
- When the following JMS header fields are specified for the JMS message
  - JMSPriority
  - JMSExpiration

This function does not operate if transaction is rolled back by executing the setRollbackOnly method. So, if an error occurs and recovery by transaction rollback cannot be expected, use the onMessage method to return EJBException and use the message save function for an error.

Performance Option

In the EJB service, an option to improve processing performance is offered.

Each performance option works as follows.

Mass Update of Multiple Records

When the same CMP Entity Bean is updated several times during transaction, the EJB container performs batched update of multiple records to reduce the frequency of access to the database and thereby improves processing performance.

Batched update of multiple records is enabled if the following conditions are met:

- The database used and the JDBC driver must support the JDBC2.0 batched update function.

  If the database or JDBC driver used does not support the JDBC2.0 batched update function, ordinary database update processing is performed.

- The batched update function is valid only when a distributed transaction is not used.

  If a distributed transaction is used, ordinary database update processing is performed.

Caching of SQL Statements

The EJB container allows the SQL statements issued to databases to be cached within the EJB Service. By doing so, it reduces processing required for preparation of SQL statements and improves processing performance.

SQL statements are cached if the following conditions are met:

- A CMP Entity Bean is used.
- The data source used is defined using the conventional DB access environment definition and JDBC driver version JDBC1.X is selected during DB access environment definition.

  If a Symfoware database is used, define MAXSQL.

Define MAXSQL according to the following formula:

Number of SQL statements to be cached = a x 4 + b

a: Total number of CMP Entity Beans deployed to IJServer

b: Total number of the finder methods owned by CMP Entity Beans

Local invocation

This function can be used when an EJB application deployed in the IJServer is invoked only from EJB applications in the same EJB container. It is assumed in the IJServer processing that an EJB application is invoked via a network. So, EJB container processing is relaxed and the operation performance is improved with this invoke function. If an EJB application with this function specified is invoked from outside the process, an error occurs with "CORBA OBJ ADAPTER".

This function is valid when IJServer operates only EJB applications or when Web and EJB applications are operated on separate Java VMs.

Set the local call function as follows: From the Interstage Management Console, select [WorkUnit] > [IJServer name] > [EJB application] > [Application environment definition] > [Interstage extended information] and set the function at "Local call." Refer to Help for the Interstage Management Console for details of the setting procedure.

If an Entity Bean is to be invoked from outside the process when this function is set, cancel the data set to specify this function. Also, when this function is not specified, delete EJB objects with the timer. For details about this deletion, see Timer deletion of EJB object.

Setting Transaction Types and Attributes

Set the transaction type and transaction attribute in the deployment descriptor file when an EJB application is developed.

When Apworks is used for development, the deployment descriptor file needs to be set. For this purpose, use the EJB deployment descriptor editor of Apdesigner or the deployment descriptor file editor (not provided by Plus Developer) of the Component Designer.

Refer to the “Apdesigner Programmer's Guide” of Apworks or “Component Designer User's Guide” (not provided by Plus Developer) for details of the setting procedure.

The transaction management type and attribute can also be set using the Interstage Management Console. Refer to Help for the Interstage Management Console for details of the setting procedure.

Time Monitoring Functions Supported by EJB Service

The EJB Service supports the following time monitoring functions:

- EJB object idle-time monitoring function of STATEFUL Session Bean
- Timer deletion of EJB object

The table below shows the differences among functions.

Function	Monitoring	Explanation
Maximum Time Monitoring Function for Application Processing	Monitors the execution time (server processing time) of an EJB application method. If the monitoring time is exceeded, the user can specify whether to forcibly stop the application processing.	This function can detect a hang or processing delay that may occur during EJB application processing.
Waiting Time Monitoring Function for Server Return	When a reply is not received within a given time after the client sends a request to the server, a timeout is posted to the client.	This function can prevent symptoms such as a hang that may occur during EJB application processing.
EJB object idle-time monitoring function of STATEFUL Session Bean	If no business method is executed for an EJB object of the STATEFUL Session Bean even after a given period, the container deletes the EJB object corresponding to the relevant instance.	This function deletes an unnecessary EJB object that has failed to issue an ejbRemovoe method and so can optimize the operating memory.
Timer deletion of EJB object	When an Entity Bean is called from outside a process, the instance of the EJB object created by the create or finder method remains in memory unless the remove method is issued. This function automatically removes the EJB object after the lapse of a given period to the EJB object since the last access.	Automatically removing the EJB object, which remains after an Entity Bean is called from outside a process, can prevent unnecessary resources from occupying memory space.

Timeout setting of each function

The table below lists the settings of each function for time monitoring.

Function	Setting location	Default value	Maximum value	Minimum value	Remarks
Maximum Time Monitoring Function for Application Processing	Use the Interstage Management Console to set the maximum processing time of an application and whether to forcibly stop the application processing if the maximum processing time is exceeded. Refer to Help for the Interstage Management Console for details of the setting procedure.	0	86400	0	Setting 0 disables the time monitoring function. Set a number in seconds.
Waiting Time Monitoring Function for Server Return	Set the response time in period_receive_timeout in the CORBA Service operating environment file (config). Refer to “CORBA service operating environment file” in the “Tuning Guide” for details.	12 (60 seconds)	20000000	0	Setting 0 disables the time monitoring function. Set a number in seconds. Multiplying the specified value by 5 produces the actual value.
EJB object idle-time monitoring function of STATEFUL Session Bean	Make settings on the STATEFUL Session Bean application environment definition window on the Interstage Management Console. Refer to Help for the Interstage Management Console for details of the setting procedure.	1800	2147483647	0	Setting 0 disables the time monitoring function. Set a number in seconds.

Function	Setting location	Default value	Maximum value	Minimum value	Remarks
Timer deletion of EJB object	Set the EJB object timeout value on the Interstage Management Console. Refer to Help for the Interstage Management Console for details of the setting procedure.	120	2147483647	0	Setting 0 disables the time monitoring function. Set a number in seconds.

Maximum Time Monitoring Function for Application Processing

This function is used when the application operates by using the IJServer.

It can detect problems such as when the database has to wait for a long time or when an EJB application enters an infinite loop.

This section explains the processing to be performed if the maximum processing time of the application is exceeded during server processing.

This processing differs, depending upon whether a forced stop for the existing application is set or not.

- Where a forced stop is set for an existing process, a rollback is done for the transaction internally after doing the forced stop on the process where the transaction is in session.

  When the maximum processing time is exceeded, the following messages are output to the event log of the server.

  extp: ERROR: EXTP4365: Application processing time exceeded the observation time

  The following exception is reported to the client:

  java.rmi.RemoteException:CORBA UNKOWN

- Where a forced stop is not set for an existing process, the forced stop of the process where the application exists is not done.

  When the maximum processing time is exceeded, the following messages are output to the event log of the server.

  extp: WARNING: EXTP4366: Application processing time exceeded the observation time

  The client is not notified.

The following explains the processing to be performed when the client issues a processing request to the server after the maximum processing time is exceeded.

The processing differs, depending upon whether a forced stop for the existing application is set or not.

- Where a forced stop is set for a process that the application existed, processing cannot be executed. It is not output to the event log of the server.

  The following exceptions are notified to the client:

  java.rmi.RemoteException: CORBA NO_IMPLEMENT

- Where a forced stop is not set for a process that the application existed, processing is executed as usual.

Waiting Time Monitoring Function for Server Return

The CORBA Service has a time-out watch function to observe the operation of applications. One of the operations observed is the time between the issue of the method of the server by the client and the receipt of the method by the client. This function can be used In the EJB service.

Refer to CORBA Application Timeout Monitoring.

If the wait time before the server method returns to the client exceeds the specified value, communication between the server and client is cut off and the following message is output to the event log of the client.

OD: ERROR: od10925:Client timeout.

The following exception is reported to the client:

java.rmi.MarshalException: CORBA COMM_FAILURE

If, a request of processing is made from the client to the server after the wait time before the server method returns to the client exceeds the specified value, retry starting with the create method. Because there is a possibility that a Session Bean remains, remove such a Session Bean using the session timeout.

Idle-time monitoring function of STATEFUL Session Bean

With the idle-time monitoring function of STATEFUL Session Bean enabled, the container deletes the EJB object corresponding to the relevant instance if a business method is not executed for the EJB object of the STATEFUL Session Bean after the lapse of a given time.

This function deletes unnecessary EJB objects for which the ejbRemove method was not issued so that the memory can be reused. The default is 30 minutes.

If a request is issued from a client for the instance corresponding to the EJB object that has been deleted, the container returns one of the following exceptions depending on the type of the interface via which access is made.

Via-interface	Exception returned
Remote interface	java.rmi.NoSuchObjectException is returned to the client.
Local interface	javax.ejb.NoSuchObjectLocalException is returned to the client.

Note

- When a timeout occurs, the container calls the ejbRemove method. It automatically removes the EJB object even when an exception occurs in the ejbRemove method.
- When a timeout occurs while the transaction management type is "Bean," the container calls the ejbRemove method. If transaction processing is in progress, the container automatically rolls back the transaction.

Setting Values for Individual Time Monitoring Functions

When two or more time monitoring functions supported by the EJB service are used concurrently, note the following requirements for setting the times for individual monitoring functions.

T(s) > T(c) > T(a)

Where:

T(s) indicates the EJB object idle-time monitoring function of STATEFUL Session Bean

T(c) indicates the waiting time until the server method returns to the client

T(a) indicates the maximum processing time of the application

Timer deletion of EJB object

This section explains the timer deletion function of EJB object.

Use of Rapid Invocation

The instance of EJB object generated by the create/finder method disappears from memory when the remove method is issued.

The instances of EJB object generated by the create/finder method remain in the memory when specifying Entity Bean as a Rapid Invoking Bean because Entity Bean usually does not call remove method.

To prevent this situation, there is a function to delete the instances of leftover EJB objects from the final access to Entity Bean after a fixed time. This function is called, "Timer deletion function of EJB object", and the timer setting is called, "EJB object time-out value of Entity Bean".

When an Entity Bean is called from inside the process, the EJB object is subjected to Java garbage collection and deleted automatically at one of the following events.

- When a Bean that calls an Entity Bean is removed.
- When the STATEFUL Session Bean no-communication monitoring function for a Bean (Session Bean) that calls an Entity Bean detects a timeout.
- When the time-out of EJB object of a Bean (For Entity Bean) that calls an Entity Bean is generated

In this way, the timer deletion function for EJB objects is useful for Entity Beans that are called from outside a process.

Set the Entity Bean EJB object timeout value by selecting [WorkUnit] > [IJServer name] > [EJB application] > [Application environment definition] > [Interstage extended information] from the Interstage Management Console. Refer to Help for the Interstage Management Console for details of the setting procedure.

Notes

Set the EJB object time-out to a higher value than the transaction time-out value. This is so that EJB object is not deleted in the transaction.

In the case of calling Entity Bean outside of process, EJB object remains in the memory while the timer is not deleted and/or a large amount of records of the database needs to be treated because of frequent accesses via CORBA communication route. Therefore, set Entity Bean for the same JavaVM if it is possible.

Notes in EJB Service

When using EJB applications by deploying them on IJServer, note the following:

- The name of the EJB application to be deployed must not exceed 255 characters..
- The same EJB application cannot be deployed to two or more IJServers (except an IJServer for which servlet and EJB run on the same JavaVM).
- If operation of a Message-driven Bean cannot be continued because the event service has stopped, JavaVM stops.
  Therefore, other EJB applications deployed to the same IJServer as the Message-driven Bean in which an error occurred also stop.
- If the EJB applications deployed to the IJServer use the same remote or home interface, an error occurs when they are started (except when a local call is used or for an IJServer for which Servlet and EJB run on the same JavaVM).
- When EJB application methods are called from a client, the number of requests from the client may exceed the specified maximum number of threads that can be processed (the default is 64). If so, the requests from the client are put in the serial queue in units of IJServer.
- When a distributed transaction is used, two or more IJServer processes cannot be started concurrently.

Chapter 9

EJB Application Development

Application Development Flow

This section describes a series of operations from developing applications to debugging applications using the Interstage Management Console.

After completion of application debugging, start IJServer to run EJB applications.

Using Apworks, which is a component-oriented integrated development support tool by Fujitsu, enables user-friendly view operation that integrates a series of procedures.

Apworks provides the development support function for various EJB applications and client applications and so can improve application development productivity.

Developing an EJB Application

- When EJB applications are linked between servers, package the EJB application of each server.
- To operate IJServer as multiple processes when Message-driven Beans and other Enterprise Beans are used, package the following separately:
  - Message-driven Beans, and Enterprise Beans that are called only from Message-driven Beans (operated in only one process)
  - All other examples (operated in multiple processes)

Deployment of an EJB Application

Refer to Help for the Interstage Management Console for details of deployment using the Interstage Management Console.

Debugging an EJB Application

Use the IJServer debug function to debug EJB applications.

Refer to "Debugging Applications" for details of the IJServer debug procedure.

Using the Development Environment of Other Companies

Even when using the development environments of other companies, there are no marked differences in the procedures used, from the development stage right through to operation.

Work Procedure

The work procedure from development to operation in an development environment from another company has no big differences from that in the development environment provided by Fujitsu.

In the development environment of another company, follow the procedure described in Developing an EJB Application to perform the required steps up to application packaging.

For information on the subsequent work steps, see Deployment of an EJB Application and subsequent sections.

Developing CMP Entity Beans

To develop CMP Entity Beans, deploy an EJB application and set the CMP definition using the Interstage Management Console. Refer to Help for the Interstage Management Console for details of the setting procedure.

Storage Place of Sample Applications

EJB application samples are provided in the following products:

- Interstage Application Server Enterprise Edition
- Interstage Application Server Standard Edition

The samples for calling an EJB application using a language other than Java are stored in the following locations:

C:\Interstage\EJB\sample

/opt/FJSVejb/sample

Chapter 10

How to create Entity

CMP Definitions

In addition to the six class files that must be created for a CMP Entity Bean, the two items of information shown in Table 10-1 must be defined to enable database access.

With the CMP definitions, database manipulation statements no longer need to be specified in an Enterprise Bean of CMP.

Table 10-1 CMP Definitions

Definition Information	Description
CMF Mapping definition	This defines the correspondence between a persistent field and a database column of an Entity Bean. This persistent field is called a Container-managed field (CMF).
Finder definition (CMP1.1)	This defines the finder method retrieval conditions in the SQL WHERE clause. The retrieval conditions can be used for the retrieval of a single table, and for retrieval using the ORDER BY clause.
CMR Mapping definition (CMP2.0)	When the CMP2.0 type is used, databases can be related between tables as the relationship between Beans/objects can be defined.
Query Definition (CMP2.0)	A statement which uses the FROM and SELECT (WHERE) clauses to search one or more EJB objects that can be defined as an EJB QL statement.

Definition Method

For CMP1.1

Set a CMP definition as follows: From the Interstage Management Console, select [WorkUnit] > [IJServer name] > [EJB application] > [Application environment definition], and set data at [CMF mapping definition] and [finder definition].

CMF mapping definition on the Interstage Management Console is used to associate database columns with persistence fields. They can be associated even when Apworks is used.

Refer to Help for the Interstage Management Console for details of the setting procedure.

When Apworks is used to develop EJB applications, the Apdesigner EJB deployment descriptor editor of Apworks or the deployment descriptor file editor (not provided by Plus Developer) of the Component Designer can also be used for CMP definition.

Refer to the “Apdesigner Programmer’s Guide” of Apworks or “Component Designer User’s Guide” (not provided by Plus Developer) for details of the setting procedure.

For CMP2.0

Set a CMP definition as follows: From the Interstage Management Console, select [WorkUnit] > [IJServer name] > [EJB application] > [Application environment definition], and set data at [CMF mapping definition] and [Query definition].

Make a query definition in advance during creation of an EJB application.

Refer to Help for the Interstage Management Console for details of the setting procedure.

Notes on Instance Management Modes

An instance management mode can be selected in an Entity Bean runtime environment, according to the user's purpose. Note the points in Table 10-2, according to the instance management mode to be selected.

Table 10-2 Notes on Instance Management Modes

Note	Instance Management Mode	Action
If the create and remove methods are issued by an application that invokes an Entity Bean, an error occurs. Even if the value of a persistent field is updated, it is not reflected in the database.	ReadOnly	When update processing is to be performed, specify a mode other than ReadOnly.
When "Cannot be operated (false) " is set to the reentrant attribute of deployment descriptor, and the method of the EJB application does the reentrant call, it operates normally.	ReadOnly	When the reentrant call to be invalid, specify a mode other than ReadOnly.
When the distributed transaction is used, and Oracle database is used, then execute retrieving multiple Instances (Enumeration or Collection type) causes “ORA-01002: Invalid fetch order” is returned.	Sequential	Specify ReadWrite when update processing is done, and Readonly when retrieval processing is done.
When the transaction attribute is specified as other than Mandatory, then execute retrieving multiple Instances (Enumeration or Collection type) may cause exception.	Sequential	Specify the transaction attribute for each EJB application (each Bean) then execute it.
Sequential cannot be set for the instance management mode of the following EJB applications: Entity Bean that is deployed to IJServer and for which "No" is specified for "Local call" on the Interstage Management Console Entity Bean that is deployed to IJServer and for which "No" is specified for "Local call" in customization	Sequential	To run the Entity Bean with Sequential specified for the instance management mode, set "Yes" for "Local call." To perform operation with "No" specified for "Local call," change the instance management mode as follows: To perform update processing, specify ReadWrite. To perform only retrieval processing, specify ReadOnly.

Correspondence between Data Types Defined in a CMP, and DBMS SQL Data Types

The following data types defined in a CMP are associated with the SQL data types of DBMS by the container:

- Data type of CMF
- Data type of parameters of the finder method

The following chapter describes the correspondence between these data types and the SQL data types of DBMS.

The following data types are supported:

- Standard data types
- Other data types.

Standard Data Types

Available Standard Data Types

The standard data types that can be used are shown in the following:

- boolean
- java.lang.Boolean (note)
- byte
- java.lang.Byte (note)
- byte[]
- char
- java.lang.Character (note)
- double
- java.lang.Double (note)
- float
- java.lang.Float (note)
- int
- java.lang.Integer (note)
- long
- java.lang.Long (note)
- short
- java.lang.Short (note)

- java.lang.String
- java.math.BigDecimal
- java.sql.Date
- java.sql.Time
- java.sql.Timestamp

Note

Will be converted into the corresponding primitive data types of Java and set in the database by the container.

CMF Data Types for which Null Values Can be Used

The data types for which database null values can be used in a CMF are as follows:

- java.lang.Boolean
- java.lang.Byte
- byte[]
- java.lang.Character
- java.lang.Double
- java.lang.Float
- java.lang.Integer
- java.lang.Long
- java.lang.Short
- java.lang.String
- java.math.BigDecimal
- java.sql.Date
- java.sql.Time
- java.sql.Timestamp

Recommended Data Types

The recommended combinations of Java data types and the DBMS SQL data types are shown in the following Table.

When using combinations other than those shown below, make sure they conform to the conversion rules for the JDBC driver.

Nevertheless, regarding the data types with (Note) in the “Available Standard Data Types”, however, obey the conversion rules of JDBC with the primitive data type of Java after conversion by the container.

Interstage EJB supports the data type within the range of JDBC2.0.

Note

When DBMS uses CHAR type columns with Oracle, data incompatibility might occur if the Oracle JDBC driver deletes the blank (space) after the returned character string or does padding to match the string length..

This is likely when you map a CHAR type column to a primary key field.

This problem is solved by using VARCHAR2 type instead of CHAR type.

Table 10-3 Recommended Data Types

Data Type of Java	SQL Data Type of DBMS (Symfoware)	SQL Data Type of DBMS (Oracle)	SQL data type of DBMS (SQL Server)
boolean/java.lang.Boolean	-	NUMBER	bit
byte/java.lang.Byte(*1)	-	NUMBER	tinyint
char/java.lang.Character	-	-	-
double/java.lang.Double	FLOAT DOUBLE	NUMBER	float
float/java.lang.Float	REAL	NUMBER	real
int/java.lang.Integer	INT	NUMBER	int
long/java.lang.Long	-	NUMBER	bigint
short/java.lang.Short(*1)	SMALLINT	NUMBER	smallint
java.lang.String	CHAR NCHAR VARCHAR NVARCHAR	CHAR(*3) VARCHAR2(*3) LONG(*3)	char varchara text nchar nvarchar sql_variant sysname uniqueidentifier ntext
java.math.BigDecimal	NUMERIC DECIMAL	NUMBER	decimal numeric money smallmoney
byte[]	BLOB (*2)	NUMBER RAW LONGRAW(*3)(*4)	binary varbinary image timestamp
java.sql.Date	DATE	DATE	-

Data Type of Java	SQL Data Type of DBMS (Symfoware)	SQL Data Type of DBMS (Oracle)	SQL data type of DBMS (SQL Server)
java.sql.Time	TIME	DATE	-
java.sql.Timestamp	TIMESTAMP	DATE	datetime (*5)(*6) smalldatetime(*5)

-: There is no data type to be recommended specifically.

Notes

1. When a CMP2.0 method is executed using the Orcle8i JDBC driver while the CMF Java data type is byte/java.lang.Byte" or "short/java.lang.Short," an error with the message "Invalid column type" may occur.

   In this case, change the CMF Java data type to "int/java.lang.Integer”or“long/java.lang.Long."

2. Fetch and update operations cannot be performed for a BLOB type column that equals or exceeds 32KB under the following conditions:

   - Symfoware with a version/level earlier than V5.0L10 is used.
   - Symfoware with version/level V5.0L10 or later is used in linkage with RDA-SV.

3. The data sizes that can be handled for Oracle data types are limited as shown below:

   - 4000 bytes
   - 4000 bytes
   - 4000 bytes
   - 2000 bytes

   To handle more data than the above limits, use BMP Entity Beans.

4. If Mandatory is specified for the transaction attribute of a CMP Entity Bean while the Oracle LONGRAW data type is used, the Oracle JDBC driver may output the following message:

   "ORA-17027: The stream has already been closed"

   Specifying Required for the transaction attribute can solve the above problem but may cause the processing performance to deteriorate.

5. Please specify data in the form of the following when you update the data of the datetime type and the smalldatetime type.
   - Description form

     YYYY-MM-DD hh:mm:ss

     YYYY:year,MM:Month,DD:Day,hh:Hour,mm:Minute,ss:Second

     The millisecond is unsupported.

     Example:

     2001-09-22 14:23:40
6. When the following API is used for the datetime type, only the value within the range of the smalldatetime type becomes effective.
   - PreparedStatement.setTimestamp(int parameterIndex, Timestamp x)

Other Classes

Classes that Can be Defined

For CMP1.1, the following classes can be defined:

- Class that directly or indirectly implements the java.io.Serializable interface
- Array of the above class.

If these classes are defined, convert them into byte[] and set them to DBMS. Null can also be specified. For mapping between byte[] and the SQL data types of DBMS, refer to Standard Data Types.

Note

Data types of the Home interface and Remote interface cannot be used in a CMF.

Using the Development Environment of Other Companies

Even when using the development environments of other companies, there are no marked differences in the procedures used, from the development stage right through to operation.

Work Procedure

The work procedure from development to operation in an development environment from another company has no big differences from that in the development environment provided by Fujitsu.

In the development environment of another company, follow the procedure described in EJB Application Development to perform the required steps up to application packaging.

For information on the subsequent work steps, see Deployment of an EJB Application and subsequent sections.

Developing CMP Entity Beans

To develop CMP Entity Beans, it is necessary to use the Interstage Management Console to set CMP definitions after the EJB application is installed. A detailed description of this procedure is provided in the Interstage Management Console Help.

Storage Place of Sample Applications

The EJB sample application is provided in the following products.

- Interstage Application Server Enterprise Edition
- Interstage Application Server Standard Edition

The sample for invoking an EJB application from a language other than Java is stored in the following directory:

Windows

C:\Interstage\EJB\sample

Solaris OE Linux

/opt/FJSVejb/sample

Chapter 11

How to call EJB Applications

This chapter explains how to call EJB applications.

To call an EJB application from a client application, use one of the following client application interfaces:

- Home interface
- Remote interface
- LocalHome interface
- Local interface.

Calling Session Beans

Calling procedure

Create a client application to call a Session Bean as follows:

1. Search for the Home interface

   Perform lookup processing to let the Naming Service inquire for the location of the Session Bean object. Refer to "Referencing Objects" in Chapter 3 for details of lookup processing.

2. Generate a Session Bean instance

   Use the create method defined for the Home interface to generate a Session Bean instance.

3. Call the business method

   Call the business method defined for the Session Bean Remote interface and perform necessary processing.

4. Delete the Session Bean instance

   Call the remove method defined for the Session Bean Remote interface to delete the Session Bean instance.

Note

The EJB application that calls the STATEFUL Session Bean may be used for transaction operation. In this case, be sure to wait for completion of the transaction processing that is initiated at execution of the business method, and then release (remove method) the STATEFUL Session Bean.

If release processing is performed during transaction processing, the following errors occur:

- javax.ejb.RemoveException is returned to the calling source.
- The following error message is output to the event log or system log:

  EJB: Error: EJB1061: Transaction is in progress.

Calling Entity Beans

Using an Entity Bean enables database access without recognizing the database operation languages such as SQL.

The client application that calls Entity Beans need not distinguish BMP and CMP.

Specifying search processing

This section explains the application that calls an Entity Bean whose instance is to be searched for, and a processing flow between the container and Enterprise Bean class. The section also provides an example of specifying the application that calls Entity Beans.

Example of searching for one instance

Outline of processing to be specified

1. Perform lookup processing for the Entity Bean to be called to obtain the EJB Home of the Entity Bean.
2. Call the findByPrimaryKey method to obtain the primary key object.
3. Call the business method.

Example of searching for multiple instances (collection interface)

Outline of processing to be specified

1. Perform lookup processing for the Entity Bean to be called to obtain the EJB Home.
2. Call the find <METHOD> method to obtain the primary key object.
3. Call the business method.

Note

If "Mandatory" is set for the transaction attribute in CMP1.1 or BMP mode in which the distributed transaction function is not used, the following methods cannot be used with the Collection interface returned from the finder method:

- add(Object o)
- addAll(Collection c)
- clear()
- contains(Object o)
- containsAll(Collection c)
- remove(Object o)
- removeAll(Collection c)
- retainAll(Collection c)
- size()
- toArray()
- toArray(Object[] a)
- The iterator() method is executed twice or more for the same Collection.

The size method cannot be used under the following conditions. If it is used, java.lang.UnsupportedOperationException is returned.

- Sequential is specified for the instance management mode.
- Mandatory is specified for the transaction attribute.
- Interstage V3 processing mode is specified.
- An EJB application deployed under Interstage V3 or earlier is used.

Relationship between Enterprise Bean Instance, EJB Object, and EJB Home

This section explains the relationship between the Enterprise Bean instance, EJB object, and EJB home.

The client application first obtains an EJB home object reference from the Naming Service. An EJB home method (such as a create or finder method) is executed to obtain an EJB object reference. When the method is executed for the EJB object, the container passes processing to the Enterprise Bean instance as needed.

EJB object and Enterprise Bean instance generation timing

The timing for generating EJB objects and Enterprise Bean instances and the number of these objects and instances generated vary depending on the type of Enterprise Bean.

STATELESS Session Bean

Because the STATELESS Session Bean holds no transaction status information and application variables in Enterprise Bean instances, Enterprise Bean instances are pooled for operation.

When a request is received from the client, the EJB container fetches one instance from the pool and executes processing on the instance. After completion of processing, the container returns the instance to the pool and returns the processing results to the client.

At the first access to the process, the number of STATELESS Session Bean instances generated is the same as the number specified for concurrent connections. In addition, only one EJB object that passes a client request to an Enterprise Bean instance is generated at activation.

The EJB object and Enterprise Bean instance are deleted when IJServer is stopped.

STATEFUL Session Bean

A STATEFUL Session Bean can hold transaction status information and application variables in an Enterprise Bean instance. Every time a create method is executed for EJB home, an EJB object and Enterprise Bean instance are generated. When access is made to the same EJB object, processing is performed on the same Enterprise Bean instance. After completion of processing, a remove method is executed for the EJB object to delete the EJB object and instance.

Entity Bean

An Entity Bean EJB object is generated by the container as needed when a method is executed for the EJB home method.

As many Entity Bean instances as specified for the initial instance count are generated and pooled. The timing for generating Enterprise Bean instances varies depending on the instance generation mode. Unlike the positioning of the Enterprise Bean instances of the STATELESS Session Beans, Entity Bean Enterprise Bean instances are mapped to DBMS records. As many Enterprise Bean instances as there are DBMS records accessed by one client (one transaction) are fetched from the pool and used. After completion of the transaction, the Enterprise Bean instances are returned to the pool.

If no Enterprise Bean instance exists in the pool, one Enterprise Bean instance used within the same transaction is selected. The record data stored in the instance is applied to the DBMS and the instance is reused as an instance that holds other record data. If no Enterprise Bean instance has been used within the same transaction, only one Enterprise Bean instance is dynamically generated and used. The Enterprise Bean instances that have been used are normally returned to the pool after completion of the transaction. However, the instance that was generated dynamically is abandoned after completion of the transaction. If Enterprise Bean instances are reused frequently, performance will be affected adversely.

Define the number of Enterprise Bean instances to be started initially (initial instance count). Refer to Help for the Interstage Management Console for details of the procedure.

Message-driven Bean

Because the STATELESS Session Bean holds no transaction status information and application variables in Enterprise Bean instances, Enterprise Bean instances are pooled for operation.

When a message is distributed from the destination, the EJB container fetches one instance from the pool and executes processing on the instance. After completion of processing, the container returns the instance to the pool and finishes processing.

Method called to generate or delete an Enterprise Bean instance

The relevant type of method for an Enterprise Bean instance is executed to generate or delete an Enterprise Bean instance.

Table 11-1 To generate an Enterprise Bean instance

Bean Type	Method Name
STATELESS Session Bean	setSessionContext ejbCreate
STATEFUL Session Bean	setSessionContext ejbCreate
Entity Bean	setEntityContext
Message-driven Bean	setMessageDrivenContext ejbCreate

Table 11-2 To delete an Enterprise Bean instance

Bean Type	Method Name
STATELESS Session Bean	ejbRemove
STATEFUL Session Bean	ejbRemove
Entity Bean	unsetEntityContext
Message-driven Bean	ejbRemove

For the STATELESS Session Bean, Entity Bean, and Message-driven Bean, each respective method shown above is called to delete an instance when the bean is stopped.

In forced stop mode, however, the instance is forcibly stopped without method execution.

Using Java Applets

This section explains the procedure for developing a Java applet that calls EJB applications.

Using Portable ORB

When a client application is developed as a Java applet, the EJB Service permits the use of the Portable-ORB.

The Portable-ORB has the following features.

- Applicable to the Internet and intranet

  Because the Portable-ORB can be downloaded from the Web Server, Thin clients such as network computers and mobile terminals can be used as Interstage clients.

- Excellent operability and maintenance

  The Portable-ORB need not be installed in advance on individual client terminals and therefore can reduce the operation and maintenance costs of clients.

Note

When the Portable-ORB is used with the EJB Service, use JBK plug-ins. JavaVM and Java plug-ins of browsers cannot be used. For details of JBK plug-ins, refer to the “Apworks J Business Kit Online Manual”.

Development procedure (pre-installed version Java library)

This section describes development procedures for the pre-installed Java library.

Descriptions of HTML Files

To run an applet, specify the applet in a HTML file using the <applet> tag.

Note

- Use a JBK plug-in. Browser JavaVMs and Java Plug-ins cannot be used.
- To install Netscape Navigator/Communicator on a machine on which JBK is already installed, manually store the start DLL in the Netscape plug-in directory. Refer to the “Apworks J Business Kit Online Manual” for details.

Applet Programming

Developing a client application as a Java applet using an EJB client differs from developing a Java application in the following point.

Lookup processing for inquiring of the Naming Service for the location of EJB application object. Make a class declaration with the class name specified in the <applet> tag in the HTML file.

An example of specifying lookup processing for a Java applet is shown below:

Example

Using the JBK Plugin

```
<HTML>
<HEAD><!--demo.html-->
<TITLE>Java sample Applet </TITLE>
</HEAD>
<BODY>
<OBJECT CLASSID="CLSID:BEA62964-C40B-11D1-AACA-00A0C9216A67"
WIDTH=300 HEIGHT=250>
<PARAM NAME="TYPE" VALUE="application/x-JBK-Plugin">
<PARAM NAME="CODE" VALUE="Sample.class">
<COMMENT>
<EMBED TYPE="application/x-JBK-Plugin"
  NAME="Sample" CODE="Sample.class" WIDTH=300 HEIGHT=250>
</EMBED>
</COMMENT>
</OBJECT>
</BODY>
</HTML>
```

Using the JBK Plugin(Jar form archive file)

```
<HTML>
<HEAD><!--demo.html-->
<TITLE>Java sample Applet </TITLE>
</HEAD>
<BODY>
<OBJECT CLASSID="CLSID:BEA62964-C40B-11D1-AACA-00A0C9216A67"
WIDTH=300 HEIGHT=250>
<PARAM NAME="TYPE" VALUE="application/x-JBK-Plugin">
<PARAM NAME="CODE" VALUE="Sample.class">
<PARAM NAME="ARCHIVE" VALUE="Sample.jar">
<COMMENT>
<EMBED TYPE="application/x-JBK-Plugin"
  CODE="Sample.class" ARCHIVE="Sample.jar" WIDTH=300 HEIGHT=250>
</EMBED>
</COMMENT>
</OBJECT>
</BODY>
</HTML>
```

```
import java.awt.*;   //Abstract window tool kit class
public class Sample extends java.applet.Applet   //Declaration of applet class
{
…
// InitialContext
Hashtable env = new Hashtable();                                          (1)
env.put("java.naming.factory.initial",
        "com.fujitsu.Interstage.ejb.jndi.FJCNCtxFactoryForClient");       (1)
```

```
env.put("java.naming.applet", this);                                    (1)
javax.naming.Context ic = new javax.naming.InitialContext( env);        (2)
// lookup
java.lang.Object Obj = (java.lang.Object)ic.lookup("SampleBean");       (3)
// home narrow()
h = (SampleHome)javax.rmi.PortableRemoteObject.narrow( Obj, SampleHome.class);
(4)
```

(1) Specify environment information required to create a context. Specify as shown in the above example.

(2) Create a Context used to perform lookup. Specify as shown in the above example.

(3) Perform lookup. Specify an EJB application name as an argument. If the lookup fails, a “javax.naming.NameNotFoundException” exception will occur. The cause of the error is displayed in a detail message of the exception concerned.

Refer to “Messages” for details of the action to be taken when lookup fails.

Perform narrow processing for the object looked up. Issue javax.rmi.PortableRemoteObject.narrow.

Notes

- Perform the operation to obtain InitialContext (above 1 and 2) in the constructor, so that it is performed only once in an application.
- See "Setting client environment" for the procedure for setting pre-installed version environment variables.
- Match the Java applet file name with the class name declared in the applet (distinguish uppercase and lowercase).

Including Java class files in a jar File

Include the following class files in a jar file:

- Applet

Bundling multiple class files in a single jar file can shorten the time taken to download these files from the Web Server.

Packaging an Applet as a jar File

An applet created with Apworks is automatically packed in a jar file. For details of applet development using Apworks, refer to the “Apworks Apdesigner Programmer's Guide” or “Component Designer User's Guide” (note that these manuals are not provided for Plus Developer).

If Apworks is not used, use the *jar* command to package an applet as a jar file.

Using the *jar* Command

For information on how to use the *jar* command, refer to the JDK documents.

An example of the *jar* command is shown below.

Example

Windows

```
jar cvf SampleApplet.jar *.class samplepkg\*.class
```

Solaris OE Linux

```
jar cvf SampleApplet.jar *.class samplepkg/*.class
```

Specify the name of the jar file to be created and the class files to be put in the jar file. The created jar file contains the files in subfolders.

Note

When the applet is used without using Portable-ORB, it is not possible to use it by downloading the client distribution data of the EJB application used from the Web server. To use it, copy the client distribution data in the client environment and set the directory of the copy destination to CLASSPATH.

Client Setup (Pre-installed Java Clients)

Setting Permission for Java Libraries

This setting is necessary when JDK/JRE 1.2.2 or later is used. When executing a Java applet, set permission for the Java library.

The method for setting permission for Java libraries using PolicyTool (attached to JDK) is described below.

Use the following procedure:

1) Start PolicyTool.

2) Press the Add Policy Entry button on the [PolicyTool] screen that is displayed after startup.

3) Enter the following on the [Policy Entry] screen:

Item	Set Value
CodeBase:	<For JDK1.3> file:<CORBA client installation directory>/etc/class/ODjava2.jar (Note 1) file:/C:/Interstage/JDK13/jre/lib/- <For JRE1.3> file:<CORBA client installation directory>/etc/class/ODjava2.jar (Note 1) file:/C:/Interstage/JRE13/jre/lib/- <For JDK1.4> file:<CORBA client installation directory>/etc/class/ODjava4.jar (Note 1) file:/C:/Interstage/JDK14/jre/lib/- <For JRE1.4> file:<CORBA client installation directory>/etc/class/ODjava4.jar (Note 1) file:/C:/Interstage/JRE14/jre/lib/-
signed by:	(None)

Note 1: Use “/” as a separator.

4) Press the Add Permission button on the [Policy Entry] screen.

5) Enter a value for each field on the [Permissions] screen, as shown in the following table.

To set permission from the [Permissions] screen, enter values for the [Permission:/Target Name:/Actions:] fields and click the OK button. At this point, control is returned to the [Policy Entry] screen.

To set another permission, click the Add Permission button again from the [Policy Entry] screen. Repeat this process and enter the required information.

Click the Done button on the [Policy Entry] screen after all of the values have been entered.

Table 11-3 Permission necessary for usual operation

These permissions ensure security during usual operation.

Permission type	Setting Permission		
	Permission	Target Name:	Actions:
Run time permission	RuntimePermission	loadLibrary.DLL name (Note 2)	Setting not required
Property permission	PropertyPermission	com.fujitsu.*	read

Note 2: The following dynamic link library (DLL) names are specified depending on the function to install when JDK/JRE is used. The extension need not be specified.

Function to Install	JDK/JRE	Dynamic Link Library
CORBA Service Client (Client Function)	JDK/JRE1.3.1	ODjava2
	JDK/JRE1.4.0	ODjava4
CORBA Service (Server Function)	JDK/JRE1.3.1	ODjavas2
	JDK/JRE1.4.0	ODjavas4

Table 11-4 Permission necessary to collect internal logs of CORBA service (Note 3)

Permission type	Setting Permission		
	Permission	**Target Name:**	**Actions:**
Property permission	PropertyPermission	user.dir	read
		java.class.path	read
File permission	FilePermission	${user.dir}*	read, write
		%OD_HOME%\etc\ config (Note 4)	read

Note 3: Refer to “config” in the “Tuning Guide” for details of an internal log of the CORBA Service. Delete the added permission after collecting logs.

Note 4: %OD_HOME% specifies the installation directory of the CORBA Service or the CORBA Service client. Default is C:\Interstage\ODWIN.

- Select File | Save from the menu bar.
- Close PolicyTool by selecting File | Exit from the menu bar.

Note

At the initial execution of PolicyTool, select [File]->[Save As] from the [PolicyTool] menu bar before termination of PolicyTool, and specify the name and storage location of the policy file. Refer to Digital Signature of JDK/JRE1.3 or later (when using keytool/jarsigner/policytool) for details of specifying policy files.

Development procedure (Portable-ORB)

This section describes development procedures for the pre-installed Java library.

Specification in the HTML file

To execute applets, specify the applets using the <applet> tag in the HTML file.

In addition, use a Portable-ORB file depending on the operating JVM.

For operation during which Portable-ORB is downloaded, specify the <APPLET ARCHIVE> or <PARM> tag (cabbase) in the HTML file used to execute the Java applet.

Files to be downloaded

The jar files to be downloaded from the Web server when an applet is executed are listed below. Digitally sign these jar files and store them on the Web server.

Theses files are individually explained below.

Applet jar files

The applet jar files include the following:

- jar file made of an applet running as a client application
- jar file consisting of client distribution data for target EJB application

jar files for Portable-ORB

Windows

Table 11-5 The names and locations of jar files for Portable-ORB are listed below

File names	Storage location
For JDK1.3 ODporbROI2_plugin.jar For JDK1.4 ODporbROI4_plugin.jar	C:\Interstage\Porb\lib

Note

The following files can also be used instead of ODporbROI2_plugin.jar:

- ODporb2_plugin.jar
- CosNaming2_plugin.jar
- InterfaceRep2_plugin.jar
- ODroi2_plugin.jar

Solaris OE Linux

Table 11-6 The names and locations of jar files for Portable-ORB are listed below

File names	Storage location
For JDK1.3 ODporbROI2_plugin.jar For JDK1.4 ODporbROI4_plugin.jar	/opt/FJSVporb/lib

Note

The following files can also be used instead of ODporbROI2_plugin.jar:

- ODporb2_plugin.jar
- CosNaming2_plugin.jar
- InterfaceRep2_plugin.jar
- ODroi2_plugin.jar

jar files for EJB Service client

The name and location of the jar file for EJB Service clients that is downloaded is provided in Table 11-7, 11-8.

Windows

Table 11-7 Names and Location of jar File for EJB Service Clients

File names	Storage location
For JDK1.3 fjcontainer72.jar For JDK1.4 fjcontainer74.jar	C:\Interstage\EJB\lib or C:\Interstage\EJBCL\lib

Solaris OE Linux

Table 11-8 Names and Location of jar File for EJB Service Clients

File names	Storage location
For JDK1.3 fjcontainer72.jar For JDK1.4 fjcontainer74.jar	/opt/FJSVejb/lib or C:\Interstage\EJBCL\lib

Note

Please use the JBK plug-in. JavaVM and Java Plug-in of a browser cannot be used.

Example

Examples of HTML file descriptions when using the pre-installed Java Library are provided below.

For JDK1.3

Using the JBK Plugin

Download the Sample.jar file using the ARCHIVE designation of the <PARAM> tag or the <EMBED> tag.

```
<HTML>
<HEAD><!--demo.html-->
<TITLE>Java sample Applet </TITLE>
</HEAD>
<BODY>
<OBJECT CLASSID="CLSID:BEA62964-C40B-11D1-AACA-00A0C9216A67"
WIDTH=300 HEIGHT=250>
<PARAM NAME="TYPE" VALUE="application/x-JBK-Plugin">
<PARAM NAME="CODE" VALUE="Sample.class">
<PARAM NAME="ARCHIVE" VALUE="SampleApplet.jar,SampleClient.jar,
        ODporbROI2_plugin.jar,fjcontainer72.jar">
<PARAM NAME="PORB_HOME" VALUE="PORBDIR">
```

```
  <COMMENT>
  <EMBED TYPE="application/x-JBK-Plugin"
    CODE="Sample.class" WIDTH=300 HEIGHT=250

ARCHIVE="Sample.jar" PORB_HOME="PORBDIR">
  </EMBED>
  </COMMENT>
  </OBJECT>
  </BODY>
  </HTML>
```

When you do not download Portable-ORB(Using the JBK Plugin)

```
  <HTML>
  <HEAD><!--demo.html-->
  <TITLE>Java sample Applet </TITLE>
  </HEAD>
  <BODY>
  <OBJECT CLASSID="CLSID:BEA62964-C40B-11D1-AACA-00A0C9216A67"
  WIDTH=300 HEIGHT=250>
  <PARAM NAME="TYPE" VALUE="application/x-JBK-Plugin">
  <PARAM NAME="CODE" VALUE="Sample.class">
  <PARAM NAME="ARCHIVE" VALUE="Sample.jar">
  <PARAM NAME="PORB_HOME" VALUE="PORBDIR">
  <COMMENT>
  <EMBED TYPE="application/x-JBK-Plugin"
    CODE="Sample.class" WIDTH=300 HEIGHT=250
    ARCHIVE="Sample.jar" PORB_HOME="PORBDIR">
  </EMBED>
  </COMMENT>
  </OBJECT>
  </BODY>
  </HTML>
```

For JDK1.4

Using the JBK Plugin

Download the Sample.jar file using the ARCHIVE designation of the <PARAM> tag or the <EMBED> tag.

```
  <HTML>
  <HEAD><!--demo.html-->
  <TITLE>Java sample Applet </TITLE>
  </HEAD>
  <BODY>
  <OBJECT CLASSID="CLSID:BEA62964-C40B-11D1-AACA-00A0C9216A67"
  WIDTH=300 HEIGHT=250>
  <PARAM NAME="TYPE" VALUE="application/x-JBK-Plugin">
  <PARAM NAME="CODE" VALUE="Sample.class">
  <PARAM NAME="ARCHIVE" VALUE="SampleApplet.jar,SampleClient.jar,
         ODporbROI2_plugin.jar,fjcontainer72.jar">
  <PARAM NAME="PORB_HOME" VALUE="PORBDIR">
  <COMMENT>
```

```
  <EMBED TYPE="application/x-JBK-Plugin"
    CODE="Sample.class" WIDTH=300 HEIGHT=250

ARCHIVE="Sample.jar" PORB_HOME="PORBDIR">
  </EMBED>
  </COMMENT>
  </OBJECT>
  </BODY>
  </HTML>
```

When you do not download Portable-ORB(Using the JBK Plugin)

```
  <HTML>
  <HEAD><!--demo.html-->
  <TITLE>Java sample Applet </TITLE>
  </HEAD>
  <BODY>
  <OBJECT CLASSID="CLSID:BEA62964-C40B-11D1-AACA-00A0C9216A67"
  WIDTH=300 HEIGHT=250>
  <PARAM NAME="TYPE" VALUE="application/x-JBK-Plugin">
  <PARAM NAME="CODE" VALUE="Sample.class">
  <PARAM NAME="ARCHIVE" VALUE="Sample.jar">
  <PARAM NAME="PORB_HOME" VALUE="PORBDIR">
  <COMMENT>
  <EMBED TYPE="application/x-JBK-Plugin"
    CODE="Sample.class" WIDTH=300 HEIGHT=250
    ARCHIVE="Sample.jar" PORB_HOME="PORBDIR">
  </EMBED>
  </COMMENT>
  </OBJECT>
  </BODY>
  </HTML>
```

Remarks

- In the example of the Portable-ORB download above, it is assumed that the Portable-ORB file is in the same directory as the HTML file. When they are not in the same directory, assume the path ARCHIVE/VALUE. When using a UNIX system as a Web server, instead of setting the path, you can substitute a link file.
- PORB HOME is specified by the <PARAM NAME> tag. Designate it as the subdirectory when searching for an operating environment file.

  Refer to Portable-ORB Operation Environment File Settings for information on specifying the Web server's document root directory/PORBDIR/etc path and storage directory.
- When an applet is executed using a JBK plug-in, the tag for JBK plug-ins must be used for coding instead of the <APPLET> tag. The method of coding varies, depending on the browser used. The above coding example is for an HTML file that can be used for either Internet Explorer or Netscape Navigator/Communicator. For details of the coding method, refer to the "Apworks J Business Kit Online Manual".

Applet Programming

Developing a client application as a Java applet using an EJB client differs from developing a Java application in the following point. Make a class declaration with the class name specified in the <applet> tag in the HTML file.

Lookup processing for inquiring of the Naming Service for the location of EJB application object

An example of specifying lookup processing for a Java applet is shown below:

```
import java.awt.*;   //Abstract window tool kit class
public class Sample extends java.applet.Applet   //Declaration of applet class
{
…
// InitialContext
Hashtable env = new Hashtable();                                          (1)
env.put("java.naming.factory.initial",
        "com.fujitsu.Interstage.ejb.jndi.FJCNCtxFactoryForClient");       (1)
env.put("java.naming.applet", this);                                      (1)
javax.naming.Context ic = new javax.naming.InitialContext( env);          (2)
// lookup
java.lang.Object Obj = (java.lang.Object)ic.lookup("SampleBean");         (3)
// home narrow()
h = (SampleHome)javax.rmi.PortableRemoteObject.narrow( Obj, SampleHome.class);
(4)
```

(1) Specify environment information required to create a context. Specify as shown in the above example.

(2) Create a Context used to perform lookup. Specify as shown in the above example.

(3) Perform lookup. Specify an EJB application name as an argument. If the lookup fails, a “javax.naming.NameNotFoundException” exception will occur. The cause of the error is displayed in a detail message of the exception concerned.

Refer to “Messages” for details of the action to be taken when lookup fails.

(4) Perform narrow processing for the object looked up. Issue javax.rmi.PortableRemoteObject.narrow.

Notes

- Perform the operation to obtain InitialContext (above 1 and 2) in the constructor, so that it is performed only once in an application.
- Match the Java applet file name with the class name declared in the applet (distinguish uppercase and lowercase).

Including Java class files in a jar File

When registering Java files on a Web server, create an archive file for the collected class files so you can download all of the files simultaneously; shortening download time. Create the archive file using the jar command (included in the Java Development Kit and commonly shortened to JDK). The jar archive created with the jar command can be used with the JBK or the Java Plug-ins.

To download the archive file (applet) of Java class files that has been created from the Web server and execute it, sign the archive file. Refer to "Digital Signature in Applets" for information on signing.

Remarks

Refer to the JDK documentation for more details about how to use the jar command.

Include the following class files in a jar file:

- Applet
- Client distribution data

Bundling multiple class files in a single jar file can shorten the time taken to download these files from the Web Server.

Packaging an Applet as a jar File

An applet created with Apworks is automatically packed in a jar file. For details of applet development using Apworks, refer to the “Apworks Apdesigner Programmer's Guide” or “Component Designer User's Guide” (note that these manuals are not provided for Plus Developer).

If Apworks is not used, use the *jar* command to package an applet as a jar file.

Bundling Client Distribution Data in a jar File

Use the *jar* command to include client distribution data, which is generated during EJB application deployment, in a jar file.

For information on how to use the *jar* command, refer to the JDK documents. An example of the *jar* command is shown below.

Command Usage Examples

This section provides command usage examples.

jar Command

Specify the name of the jar file to be created and class files to be put in the jar file. The jar file created includes subdirectory files.

```
C:\jarSample>jar cvf Sample.jar *.class Samplemod\*.class
adding: Samplemod/_SampleintfStub.class (in=1282) (out=704) (deflated 45%)
adding: Samplemod/Sampleintf.class (in=302) (out=215) (deflated 28%)
adding: Samplemod/SampleintfHelper.class (in=2175) (out=994) (deflated 54%)
adding: Samplemod/SampleintfHolder.class (in=907) (out=461) (deflated 49%)
```

Notes

- When a class file to be included in an archive file is specified, path specification information is also included in the archive file. Change the current directory to the highest-level directory in which the class file to be put in the archive file is stored, and create an archive file by specifying path information according to the class file configuration.
- When "cabbase" is set to <PARAM> tag in HTML, setting the "ARCHIVE" to the <APPLET> tag will not work.

Storing jar Files in the Web Server

Store HTML files, Java applets, and IDL generation class files in the same directory on the Web server. Store the following jar files in the Web Server:

- Applet jar file
- Client distribution data jar file
- Portable-ORB jar file
- EJB Service client jar file
- HTML file that calls an applet

The client downloads the above files from the Web Server when the browser references the server.

Figure 11-1 shows a sample structure of the files stored in the Web Server:

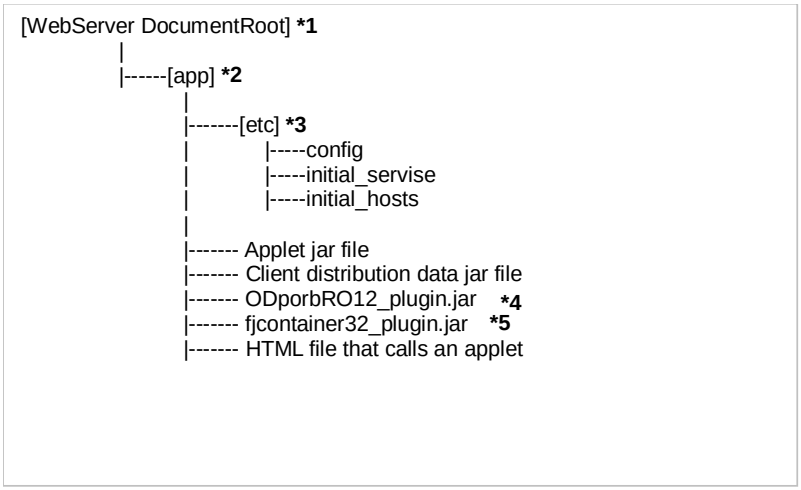


Figure 11-1 Example of Jar Files Stored in the Web Server

*1) [WebServer DocumentRoot] is a Web Server setting, which can be specified as desired by the user.

*2) [app] is the folder in which the corresponding applications are stored. You can specify any name for this folder.

*3) [etc] is the folder in which the operating environment setup files used by Portable-ORB are stored. For more information, refer to Portable-ORB Operation Environment File Settings.

*4) The Portable-ORB jar file

*5) The EJB Service client jar file

Setting up the Portable-ORB Environment in the Web Server

Define the Portable-ORB environment in accordance with the environment setup procedure used when a Portable-ORB/Java applet is downloaded from the Web Server. For this operation, refer to Portable-ORB Operation Environment File Settings.

Setting client environment (Portable-ORB)

To run a client application (applet) using Portable-ORB, make the following settings in the client environment:

- Specify the ORB (Object Request Broker)
- Portable-ORB Operation Environment File Settings
- Edit the JBK plug-in setup file

Specify the ORB (Object Request Broker)

The ORB to be used must be selected in setting up the applet execution environment.

Set the following types of Java runtime environment property information as the JavaVM start options in the jbkplugin.properties file attached to the orb.properties file or JBK plug-in.

If the same properties are written in both the orb.properties and jbkplugin.properties files, the properties written in the jbkplugin.properties file are effective.

Table 11-9 Property Names and Setting Values

Property name	Setting value
org.omg.CORBA.ORBClass	com.fujitsu.ObjectDirector.CORBA.ORB
javax.rmi.CORBA.StubClass	com.fujitsu.ObjectDirector.rmi.CORBA. StubDelegateImpl
javax.rmi.CORBA.UtilClass	com.fujitsu.ObjectDirector.rmi.CORBA. UtilDelegateImpl
javax.rmi.CORBA.PortableRemoteObjectClass	com.fujitsu.ObjectDirector.rmi.CORBA. PortableRemoteObjectDelegateImpl

Note

When operating an applet using Portable-ORB, do not use the following property:

- org.omg.CORBA.ORBSingletonClass

Portable-ORB Operation Environment File Settings

When using Portable-ORB, you need to set the PORB_HOME parameters with an HTML file in order to specify the storage position for the operation environment files. The operation environment files shown in Table 11-10 exist in Portable-ORB.

Table 11-10 Operation Environment Files

Operation Environment Files	File Name
Environment Definition File	config
Object References Information Storage File	initial_services
Object References Search Information File	initial_hosts

You need to store these operation environment files in the Web server's document root sub directory.

Note: Do not store the operation environment files under the user authentication directory when authentication is to be performed by the Web Server based on the user name and password.

There is a way to specify a different operation environment file for each applet and a way to specify the same operation environment file for multiple applets. These methods are described below: Using different operation environment file for each applet is depicted in Figure 11-2.

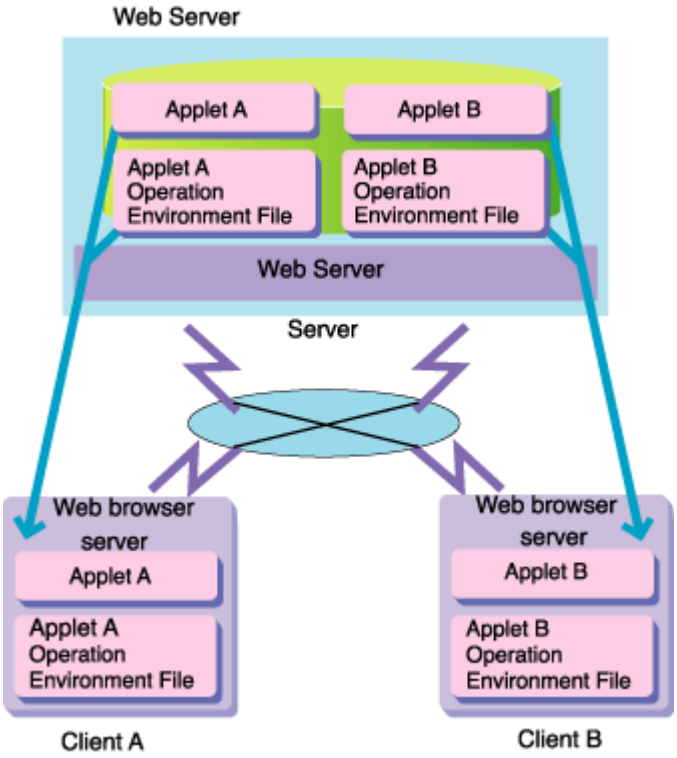


Figure 11-2 Using Different Operation Environments with Different Applets

Applet A that executes on Client A uses the Applet A operation environment file, and Apple B that executes on Client B uses the Applet B operation environment file

There are two procedures for specifying operation environment files in applet units: one that specifies PORB_HOME and another that does not specify PORB_HOME.

Specifying PORB_HOME

When specifying PORB_HOME, create an etc directory for the operation environment file of each applet and store it in its sub directory. Specify the relative path from the Web server's document root as the storage location for the operation environment files with an HTML file's <PARAM> tag in PORB_HOME parameters. The etc directory is not included in this designation.

Sample directory structures are displayed below. The operation environment file for Applet A is stored in the ap1Aenv/etc sub directory and the operation environment file for Applet B is stored in the ap1Benv/etc sub directory on the Web server's document root directory on the World Wide Web

```
/-
   - [Web] - [envfile] - [aplAenv] - [etc] – appletA operation environment file
                       - [aplBenv] - [etc] – appletB operation environment file
           - [applet ] - [appletA] – appletA.html
                                   - appletA.class
                         [appletB] – appletB.html
                                   - appletB.class
```

In the above example, the PORB_HOME parameters for each applet are set with a <PARAM> tag, shown in the following examples.

Applet A <PARAM> Tag Example

```
<HTML>
  <HEAD><!--demo.html--></HEAD>
  <TITLE>Java sample Applet </TITLE>
  <BODY>
  <H1>Java sample Applet</H1>
  <applet code="applet.class" width=300 height=250>
  <PARAM NAME=PORB_HOME VALUE=envfile/aplAenv>
  </applet><BR>
  </BODY>
  </HTML>
```

Applet B <PARAM> Tag Example

```
<HTML>
  <HEAD><!--demo.html--></HEAD>
  <TITLE>Java sample Applet </TITLE>
  <BODY>
  <H1>Java sample Applet</H1>
  <applet code="applet.class" width=300 height=250>
  <PARAM NAME=PORB_HOME VALUE=envfile/aplBenv>
  </applet><BR>
  </BODY>
  </HTML>
```

PORB_HOME Not Specified

When PORB_HOME is not specified, create the etc directory in the directory where each applet is stored and store the operation environment file there. Sample directory structures are displayed below.

```
/-
   - [Web] - [applet  ] - [appletA] – appletA.html
                                    - appletA.class
                                    - [etc] – appletA operation environment
file
                          [appletB] – appletB.html
                                    - appletB.class
```

```
                                        - [etc] – appletB operation environment
file
```

Remarks

Use the “PORB_HOME not specified” method when using the file protocol. The file protocol is used when HTML files stored on a local disk are directly specified using applet viewer (a browser and JDK tool) instead of the Web Server.

Using a Uniform Operation Environment with Multiple Applets

This is depicted in Figure 11-3.

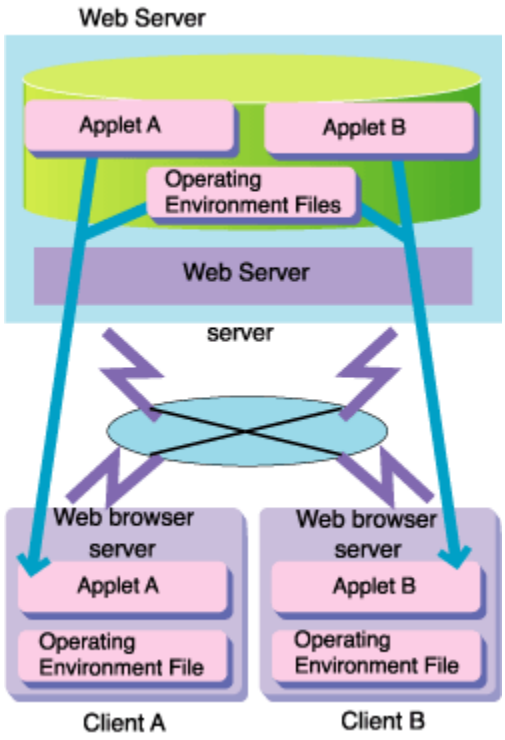


Figure 11-3 Operation Environments under Multiple Applets

Use a common operation environment file with Applet A executing on Client A and Applet B executing on Client B. For the operation environment files’ storage location, specify the relative path from the Web server’s document root with a <PARAM> tag for Applet A’s HTML file and Applet B’s HTML file in PORB_HOME parameters. The etc directory is not included in this designation.

Sample directory structures are shown below. Portable-ORB is installed on the /Web (The Web server’s document root directory) sub directory’s porb directory, and the operation environment file is stored in its sub directory’s etc directory.

```
/-
   - [Web] - [porb   ] - [lib] - ODporb.jar...
                       - [etc] – operation environment file
           - [applet ] - [appletA] – appletA.html
```

```
                                    - appletA.class
                          [appletB] – appletB.html
                                    - appletB.class
```

Specify a path to /Web/porb to be able to utilize the operation environment files on the /Web/porb/etc subdirectory in the HTML files for Applet A and B. The following examples show the <PARAM> tag descriptions for each applet.

Applet A <PARAM> Description Example

```
<HTML>
<HEAD><!--demo.html--></HEAD>
<TITLE>Java sample Applet </TITLE>
<BODY>
<H1>Java sample Applet</H1>
<applet code="applet A.class" width=300 height=250>
<PARAM NAME=PORB_HOME VALUE=porb>
</applet><BR>
</BODY>
</HTML>
```

Applet B <PARAM> Description Example

```
<HTML>
<HEAD><!--demo.html--></HEAD>
<TITLE>Java sample Applet </TITLE>
<BODY>
<H1>Java sample Applet</H1>
<applet code="appletB.class" width=300 height=250>
<PARAM NAME=PORB_HOME VALUE=porb>
</applet><BR>
</BODY>
</HTML>
```

Remarks

- Use the environment settings command *porbeditenv* for creating and/or editing operation environment files.
- When creating an etc directory to store the operating environment file, make sure you create it using lower-case alphabetical characters.

Editing the JBK Plug-in Setup File

Specify the following class paths as the JavaVM start options in the jbkplugin.properties file attached to the JBK plug-in.

- When JDK is used

  %JAVA_HOME%\jre\lib\isejb.jar

- When JRE is used

  %JAVA_HOME%\lib\isejb.jar

Setting Example

```
jbk.plugin.vmoption=-classpath C:\JBKplugin\jre13\lib\isejb.jar
-Dorg.omg.CORBA.ORBClass=com.fujitsu.ObjectDirector.CORBA.ORB
-Djavax.rmi.CORBA.StubClass=com.fujitsu.ObjectDirector.rmi.CORBA.
StubDelegateImpl
 -Djavax.rmi.CORBA.UtilClass=com.fujitsu.ObjectDirector.rmi.CORBA.
UtilDelegateImpl
-Djavax.rmi.CORBA.PortableRemoteObjectClass=com.fujitsu.ObjectDirector.rmi.
CORBA.PortableRemoteObjectDelegateImpl
```

Digital Signature in Applets

To download and operate a Java applet, put a digital signature on the applet or Portable-ORB. The Java applet running as a CORBA client is downloaded from the Web server, and accesses the remote machine on which the CORBA server application runs through a network. Therefore, a digital signature must be put on a Java applet whenever it is downloaded for operation, whether a pre-installed version Java library or Portable-ORB is used.

The jar files that require digital signature are:

- Applet jar file
- Client distribution data jar file
- Portable-ORB jar file
- EJB Service client jar file

The policy file used by the JBK plug-in can also be used to set permissions. For details of the policy file used by the JBK plug-in, refer to the “Apworks J Business Kit Online Manual”.

Digital Signature of JDK/JRE1.3 or later (when using keytool/jarsigner/policytool)

This section describes the procedures for using digital signatures. JDK signature tools such as keytool, jarsigner or policytool are used as the signature tool in this case.

Note

If operation is carried out without downloading Portable-ORB or if pre-installed Java clients are used in JDK/JRE1.3 or later, authorization needs to be set for the Java libraries in each environment, using policytool. For information about authorization setting for Java libraries, refer to Setting Permission for Java Libraries.

Remarks

Refer to JDK documentation for details about the signature tools. When using the J Business Kit, refer to the “Apworks J Business Kit Online Manual”.

Perform the digital signing procedure as follows

(1) Creating a Pair of Certificate Key

Create a pair of certificate key to specify the additional information of the certificate and the alias name to access it. The example when setting samplesigner alias name and 365 (days) valid term of the certificate is shown below.

```
keytool -genkey -alias samplesigner -dname "cn=samplesigner, ou=JAVA PROJECT,
o=FUJITSU, c=JA" -validity 365
```

-genkey

To create a pair of certificate key

-alias

Alias name to access the certificate

-dname

Signer, organization, company or country

-validity

Valid term of certificate

When executing, the passwords for the key store and the pair of certificate key to be created are required. These passwords are necessary to access to the key store and the pair of certificate key.

The key store is a database which manages information of the pair of certificate key, and it doesn't exist when JDK/JRE is installed but it is created at the first execution of keytool.

Caution

The certificates created here and the certificates to be imported by each client machine must be created at the same time. Even if the same content is specified in the -dname option, certificates created at different times will be identified as different certificates.

(2) Application to jar Archive File

Apply digital signature to jar archive file with the created certificate.

In the following example, it is signed for Sample.jar with the certificate created in step (1) above.

```
jarsigner -signedjar Sample.jar.sig Sample.jar samplesigner
```

-signedjar

Specifies the name of the jar file to which a signature is applied (the default value is the name of the signature source jar file).

Sample.jar

Specifies the name of the certificate source jar file.

samplesigner

Specifies the alias of the certificate for which a signature is executed.

When executing, the passwords for the key store and the pair of certificate key are required. Input the passwords that you specified in (1) above.

Refer to “Including Java class files in a jar File(installed version Java library)” or “Including Java class files in a jar File(Portable-ORB)” for the procedure for creating a jar archive file.

When multiple jar archive files are to be created by the same author, repeat the processing in (2).

(3) Verifying Application of a Signature in a jar Archive File

Use the following commands to verify that the signature is correctly implemented in the jar archive file that applies a signature.

```
jarsigner  -verify sample.jar.sig
```

-verify

Specifies verification of the digital signature of the jar archive file.

sample.jar.sig

Specifies the jar archive file whose digital signature is to be verified.

Normally, when a digital signature is implemented, message “jar verified” is displayed. When a digital signature is not implemented, message “jar is unsigned. (signatures missing or not parsable)” is displayed.

After checking that the digital signature is implemented normally, change the extension to *.jar to use the file as a jar file (example: delete sample.jar used before applying the signature and change sample.jar.sig to tet.jar).

To display the detailed digital signature information, execute the command by specifying the –verbose/-certs option.

(4) Exporting a Certificate

Export the certificate to be used at the client system whom an applet is downloaded into. Specify the alias name of the certificate created in (1) above.

```
keytool -export -alias samplesigner -file samplesign.cer
```

-export

To acquire the certificate

-alias

Alias name of the certificate to acquire

-file

File name to store the certificate to acquire

When executing, the password for the key store is required. Input the password that you specified in (1) above.

(5) Importing a Certificate

Import the certificate to the client system whom an applet is downloaded into. This operation needs to run at the client system. Copy the certificate, samplesign.cer to the client machine in advance.

```
keytool -import -alias sampleuser -file samplesign.cer
```

-import

To import the certificate

-alias

Alias name of the certificate to import

-file

File name to store the certificate to import

When executing, the password for the key store is required. This is required to access to the key store. When prompted to accept the certificate imported, enter "yes".

The alias (specified in –alias) option is required to specify the certificate (specified in –file) for subsequent operations. Specify an alias of the certificate for which authorization is to be set when setting policy in “(6), Setting a Permission for the Certificate.”

(6) Setting a Permission for the Certificate

Use policytool to set permission for the certificate which is imported to the client system. This operation needs to run at the client system.

The *policytool* command is a GUI tool associated to define security policies.

This command is used to set or change the authorization of Java classes signed using this command or stored in any location. For information about the settings of the *policytool* command, refer to policytool Command Setting (Supplements).

policytool Command Setting (Supplements)

Refer to “Application Interstage Application Server” for how to set the *policytool* command.

V7.0 Distributed Application Development Guide (CORBA Service Edition).

Notes

Restrictions on creating EJB applications are shown below:

- The name of the EJB application must not exceed 255 characters.
- The name of the business method must not exceed 256 characters. If the rules for conversion from Java to IDL are to be used, define a business method with a name in such a way that the business method name after conversion does not exceed 256 characters.
- The names of the Home and Remote interfaces must not exceed 234 characters (including the package name).

Chapter 12

DB Access Environment Definition

Specifying the DB Access Environment Definitions

The DB access environment definition is used to add/change/delete the datasource definition that is necessary for using the database

Note

Customize Tool is a tool for providing compatibility for environments of previous versions. Customizing with the Interstage Management Console has been possible since V6, but Customize Tool when you need the DB access environment definition.

Refer to the manual for the previous version for details of the use of Customize Tool.

Execute the following command and start Customize Tool.

```
ejbcustx -ejbdb
```

Notes

There are the following restrictions when the datasource defined in the DB access environment definition is used with IJServer.

Function	Restriction
Transaction Isolation Level	Operates at the DBMS default transaction separation level.
Pre-opened connection	Cannot be used.
The number of Maximum Connection	Operates at 64 by default.
Connection Timeout	Operates at 5 seconds by default.
Idle Timeout	Operates at 600 seconds by default.

Chapter 13

Customize by EJB Service Operation Command

This explains how to customize the EJB application execution environment and DB definition information using the operation command of Customize Tool.

Customize by the EJB service operation command allows you to edit the following contents, based on the XML format definition file that you can update: EJB application runtime environment definitions and Rapid Invocation definitions ("Enterprise Bean Definition information"), and DB access environment definitions ("DB definition information").

The EJB service operation commands to be used for customize are broadly classified into the following two functions.

- Definition File Export

  EJB application runtime environment definitions and Rapid Invocation definitions, and DB access environment definitions are exported to the XML format definition file.

- Definition File Import

  Updated XML-format definition file contents are imported to the EJB application runtime environment definitions and Rapid Invocation definitions, and DB access environment definitions.

The following describes the flow of customize using the EJB service operation command, definition file exporting and importing procedures, and the contents and description examples of each definition file.

Only information defined by the DB access environment definition becomes an export/import object for DB definition information. Refer to “DB Access Environment Definition” for details of the DB access environment definition.

Customize Flow

Customize edits each type of definition information by executing the *ejbdefimport/ejbdbdefimport* command, based on the definition file that is exported from the EJB server by using the *ejbdefexport/ejbdbdefexport* command or that is newly created.

The figure below shows the customized flow using the *ejbdefimport/ejbdbdefimport* command.

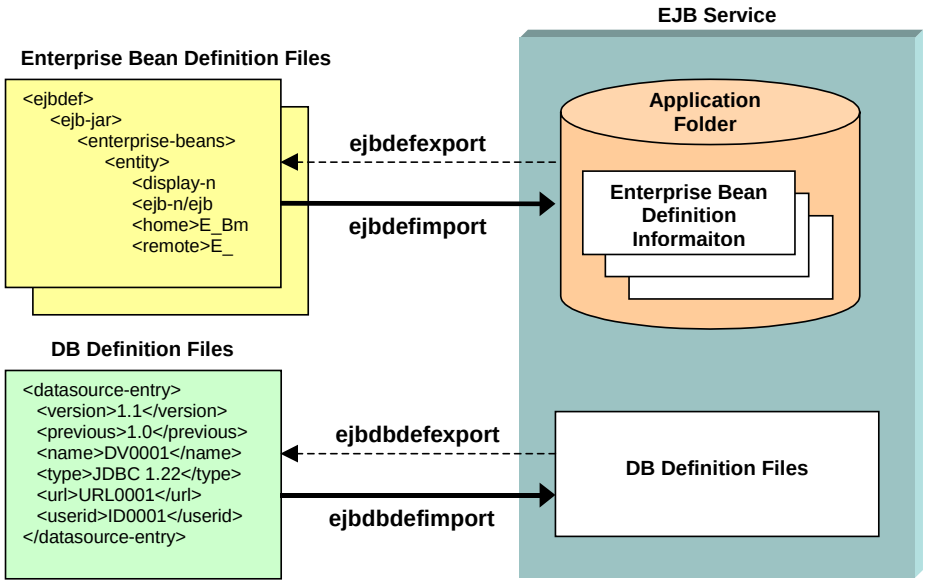


Figure 13-1 Customized Flow using the ejbdefimport/ejbdbdefimport command

This section explains each definition file and the commands that are used for customize.

The table below lists the definition files to be used to customize flow.

Table 13-1 Definition files

Definition files	Description
Enterprise Bean Definition files	This file represents the Enterprise Bean definition information (runtime environment definitions of EJB applications installed in the application folder and Rapid Invocation definitions) in XML format.
DB Definition files	This file represents the DB definition information (DB access environment definitions) in XML format.

The table below lists the command used to customize flow.

Table 13-2 Commands Used in Customize

Commands	Function
ejbdefexport	Exports the Enterprise Bean definition information to the Enterprise Bean definition file.
ejbdefimport	Imports the Enterprise Bean definition file to the Enterprise Bean definition file.
ejbdbdefexport	Exports the EJB service DB definition information to the DB definition file.
ejbdbdefimport	Imports the DB definition file to the EJB service DB definition information.

For details of each command, refer to the Reference Manual (Command Edition)

Export and Import of Enterprise Bean Definition Information

This section explains the export and import of the Enterprise Bean definition information.

Export of Enterprise Bean Definition Information

When the *ejbdefexport* command is executed, predefined Enterprise Bean definition information is exported to the Enterprise Bean definition file.

At this point, all Enterprise Bean definition files in the IJServer can be exported.

Example

- When the SampleEB information stored in the IJServer "TestIJServer" is exported

ejbdefexport SampleEB -i TestIJServer -f c:\ejb\deffile.xml

- When the information of all EJB applications installed in the IJServer "TestIJServer" is exported to the c:\ejb directory.

ejbdefexport -i TestIJServer -all c:\ejb

- When the SampleEB information stored in the IJServer "TestIJServer" is exported

  ejbdefexport SampleEB -i TestIJServer -f /tmp/ejb/deffile.xml

- When the information of all EJB applications installed in the IJServer "TestIJServer" is exported to the /tmp/ejb directory.

  ejbdefexport -i TestIJServer -all /tmp/ejb

Import of Enterprise Bean Definition Information

The Enterprise Bean definition file is imported by executing the *ejbdefimport* command. This imports the definition file contents to the Enterprise Bean definition. To prepare the Enterprise Bean definition file, create a new or edit an existing file that is exported with the *ejbdefexport* command.

Multiple Enterprise Bean definition files can be imported simultaneously.

Example

- When the Enterprise Bean definition information deployed in the IJServer "TestIJServer" defined in c:\ejb\deffile.xml is imported

  ejbdefimport -i TestIJServer -f c:\ejb\deffile.xml

- When the information of all Enterprise Bean definition files deployed in the IJServer "TestIJServer" in the c:\ejb folder is imported

  ejbdefimport -i TestIJServer -all c:\ejb

- When the Enterprise Bean definition information deployed in the IJServer "TestIJServer" defined in /tmp/ejb/deffile.xml is imported

  ejbdefimport -i TestIJServer -f /tmp/ejb/deffile.xml

- When the information of all Enterprise Bean definition files deployed in the IJServer "TestIJServer" in the /tmp/ejb folder is imported

  ejbdefimport -all /tmp/ejb

Note

If the *ejbdefimport* command is executed by specifying DB definition information in the -f option or the *ejbdefimport* command is executed while DB definition information exists in the folder specified in the -all option, the following error occurs during import.

```
An error occurred during reading of definition file (file name).
There is no root element declaration.
file:///(X) : Line X, Column X The above error was detected.
EJB3504S-20-093-XXXX
```

Export and Import of DB Definition Information

This section explains the export and import of the Enterprise Bean definition information.

Export of DB Definition Information

When the *ejbdbdefexport* command is executed, predefined DB definition information is exported to the specified file.

Example

Windows

- When DB definition information is exported to the c:\ejb\DBDef.xml file

  ejbdbdefexport -f c:\ejb\DBDef.xml

Solaris OE Linux

- When DB definition information is exported to the /tmp/ejb/DBDef.xml file

  ejbdbdefexport -f /tmp/ejb/DBDef.xml

Import of DB Definition Information

The DB definition file is imported by executing the *ejbdbdefimport* command. This imports the definition file contents to the DB definition. To prepare the DB definition file, create a new file or edit an existing one that is exported with the *ejbdbdefexport* command.

Example

Windows

- When the DB definition information, defined in c:\ejb\DBDef.xml, is imported

  ejbdbdefimport -f c:\ejb\DBDef.xml

Solaris OE Linux

- When the DB definition information, defined in /tmp/ejb/DBDef.xml, is imported

  ejbdbdefimport -f /tmp/ejb/DBDef.xml

Note

When the DB definition information is exported, the DB definition file "password" tab displays "*" as many as the password characters that are set in the DB definition information. Before importing the DB definition file exported from the EJB server, change the "password" tab value to the correct password during definition file editing.

```
An error occurred during reading of definition file (file name).
There is no root element declaration.
file:///(X) : Line X, Column X The above error was detected.
EJB3504S-20-093-XXXX
```

Contents of Enterprise Bean Definition File

This describes the Enterprise Bean definition file in XML format.

The file contents are as follows.

Table Details

Tab name: XML tab name in definition file

Value: XML value in definition file

Meaning: Tab meaning (the Customize Tool displays its item names.)

Editing: Whether values can be edited (O: Yes; X: No)

Correspondence to Interstage Management Console screen: Correspondence to customize Tool screen

Table 13-3 EJB Tab Value, Meaning and Correspondence

Tab Name			Value	Meaning	Editing	Correspondence to Interstage Management Console Screen
Session	ejb-name?		Any character string	Enterprise Bean name	X	EJB Application Information
	home?		Any character string	Home interface name	X	
	remote?		Any character string	Remote interface name	X	
	Local-home?		Any character string	Local Home interface name	X	
	local?		Any character string	Local interface name	X	
	ejb-class?		Any character string	Enterprise Bean class name	X	
	session-type?		Choose from the following values. Stateful Stateless	Session Type	X	
	transaction-type?		Choose from the following values. Bean Container	Transaction Type	O	
	env-entry+	description?	Any character string	Explanation	X	Environment Property
		env-entry-name	Any character string When the correspondence of env-entry-name is not taken in export between the definition file and Enterprise Bean definition information, it becomes an error.	Property name	X	

Tab Name			Value	Meaning	Editing	Correspondence to Interstage Management Console Screen
		env-entry-type	Choose from the following values. java.lang.Boolean java.lang.String java.lang.Integer java.lang.Double java.lang.Byte java.lang.Short java.lang.Long java.lang.Float java.lang.Charactor	Type	O	
		env-entry-value?	Choose from the following values. java.lang.Boolean:True/False java.lang.String: Any character string java.lang.Integer: -2147483648-2147483647 java.lang.Double: -1.7976931348623157E308-1.7976931348623157E308 java.lang.Byte:-128-127 java.lang.Short: -32768-32767 java.lang.Long: -9223372036854775808-9223372036854775807 java.lang.Float: -3.4028234663852886E38-3.4028234663852886E38 java.lang.Character: Any character string	Value	O	
	ejb-ref+	description?	Any character string	Explanation	X	EJB Reference
		ejb-ref-name	Any character string	Enterprise Bean JNDI name	X	
		ejb-ref-type	Choose from the following values. Session Entity	Enterprise Bean Type	X	
		home	Any character string	Home interface name	X	
		remote	Any character string	Remote interface name	X	
		ejb-link?	Any character string	Enterprise Bean name	X	

Tab Name			Value	Meaning	Editing	Correspondence to Interstage Management Console Screen
	ejb-local-ref+	ejb-local-ref-name	Any character string	Enterprise Bean JNDI name		Local EJB Reference
		ejb-local-link?	Any character string	Enterprise Bean name		
		ejb-ref-type	Choose from the following values. Session Entity	Enterprise Bean Type		
		local-home	Any character string	Local Home interface name		
		local	Any character string	Local interface name		
	security-role-ref+	description?	Any character string	Explanation	X	Security Role Reference
		role-name	Any character string	Code base Security role name	X	
		role-link	Any character string	Code base Security role name	X	
	resource-ref+	description?	Any character string	Explanation	X	Resource Reference
		res-ref-name	Any character string	Resource manager name	X	
		res-type	Any character string	Class/Interface name	X	
		res-auth	Choose from the following values. Application Container	Resource authority	X	
	resource-env-ref+	description?	Any character string	Explanation	X	Resource Environment Reference
		resource-env-ref-name	Any character string	Resource manager name	X	
		resource-env-ref-type	Any character string	Class/Interface name	X	

Tab Name				Value	Meaning	Editing	Correspondence to Interstage Management Console Screen
entity	ejb-name?			Any character string	Enterprise Bean name	X	EJB Application Information
	home?			Any character string	Home interface name	X	
	remote?			Any character string	Remote interface name	X	
	Local-home?			Any character string	Local Home interface name	X	
	local?			Any character string	Local interface name	X	
	ejb-class?			Any character string	Enterprise Bean class name	X	
	persistence-type?			Choose from the following values. Bean Container	Persistence Type	X	
	prim-key-class?			Any character string	PrimaryKey Class name	X	
	reentrant?			Choose from the following values. True False	Re-entrant type	X	
	primkey-field?			Any character string	PrimaryKey field name	X	
	env-entry+			Refer to env-entry of session tag.			
	ejb-ref+			Refer to ejb-ref of Session Tag.			
	security-role-ref+			Refer to security-role-ref of Session Tag.			
	resource-ref+			Refer to resource-ref of Session Tag			
	resource-env-ref+			Refer to resource-env-ref of Session Tag			
	query*	description?			Explanation	X	query
		query-method	method-name		method	X	
			method-params		method	X	
		result-type-mapping?		Choose from the following values. Local Remote	EJB QL	X	

Tab Name				Value	Meaning	Editing	Correspondence to Interstage Management Console Screen
message-driven-bean	ejb-name?			Any character string	Enterprise Bean name	X	EJB Application Information
	ejb-class?			Any character string	Enterprise Bean name	X	
	transaction-type?			Choose from the following values. Bean Container	Transaction Type	O	
	message-selector?			Any character string	Message Selector	O	
	message-driven-destination?	destination-type		Choose from the following values. javax.jms.Topic javax.jms.Queue	Destination Type	O	
		subscription-durability?		Choose from the following values. Durable NonDurable	Subscription Durability	O	
	env-entry+			Refer to env-entry of session tag.			
	ejb-ref+			Refer to ejb-ref of Session Tag.			
	security-identity?			Refer to security-identity of Session Tag.			
	resource-ref+			Refer to resource-ref of Session Tag			
	resource-env-ref+			Refer to resource-env-ref of Session Tag			
relationships?	description?			Any character string	Explanation	X	CMR Mapping Definition
	ejb-relation+	description?		Any character string	Explanation	X	
		ejb-relation-name?		Any character string	ejb relation name	X	
		ejb-relationship-role	description?	Any character string	Explanation	X	
			ejb-relationship-role-name?	Any character string	ejb relationship role name	X	
			multiplicity	Choose from the following values One Many	multiplicity	X	
			cascade-delete?	-	Record deletion	X	

Tab Name					Value	Meaning	Editing	Correspondence to Interstage Management Console Screen
		ejb-relationship-role	relationship-role-source	description?	Any character string	Explanation	X	
				ejb-name	Any character string	Enterprise Bean name	X	
			cmr-field?	description?	Any character string	Explanation	X	
				cmr-field-name	Any character string	cmr-field-name	X	
				cmr-field-type?	Any character string	Type	X	
			The above-mentioned reference					
security-role+	description?				Any character string	Explanation	X	Security Role
	role-name				Any character string	Security role name	X	

Tab Name			**Value**	**Meaning**	**Editing**	**Correspondence to Interstage Management Console Screen**
method-permission+	role-name+		Any character string It is necessary to define the specified character string in role-name of the security-role tag.	Security role name	O	Method Permission
	method+	description?	Any character string	Explanation	O	
		ejb-name	Any character string	Enterprise Bean name	X	
		method-intf	Any character string	Interface name	X	
		method-name	Any character string	Method name	X	
		method-params?	Any character string	Parameter	X	
container-transaction+	method+		Refer to method of method-permission Tag.		O	Transaction
	trans-attribute		Choose from the following values. NotSupported Required Supports RequiresNew Mandatory Never	Transaction Attribute	O	
description?			Any character string	Memo	O	Interstage Extended Information
version-entry?	deploy-ejb-version?		The following value fixation 1.1 2.0	Based on EJB specification version	X	
	deploy-java-version?		Choose from the following values. 1.1 1.2 1.3 1.4	JDK version used for deployment	X	

Tab Name		Value	Meaning	Editing	Correspondence to Interstage Management Console Screen
base?	jndi-name? (Note 3)	Any character string	Enterprise Bean JNDI name	X	EJB Application name
	tran-timeout?	Choose from the following values. 0-2147483647	Transaction time out	O	
	tran-kind? (Note 1)	Choose from the following values. Local Global	Distributed Transaction	O	
	local-mode?	Choose from the following values. False True	Use of local invocation	O	Interstage Extended Information
redirect? (Note 1)	redirect-mode?	Choose from the following values. False True	Standard output/Standard error output mode	O	-
	redirect-path?	Any character string	Standard output/Standard error output mode	O	
session_eb	max-instance? (Note 3)	Choose from the following values. 1−64	Number of initial start instances	O	-
	session-timeout? (Note 3)	Choose from the following values. 0−2147483647	Session timeout value	O	
	session-idle-timeout? (Note 3)	Choose from the following values. 0−2147483647	No-communicatio n monitoring time	O	
	max-ejbobject? (Note 3)	Choose from the following values. 1−64	Number of Stateful Beans that are connected concurrently	O	
	stateless-instance-create-type? (Note 3)	Choose from the following values. At Start-Up At First Access	Stateless and other bean reference function	O	
entity-eb?	entity-timeout?	Choose from the following values. 1-2147483647	EJB object time out value of Entity Bean	O	Interstage Extended Information
	entity-instance-type	Choose from the following values. ReadWrite ReadOnly Sequential	Entity Bean instance management mode	O	
	entity-instance-size	Choose from the following values. 1-2147483647	Number of Entity Bean instances	O	

Tab Name			Value	Meaning	Editing	Correspondence to Interstage Management Console Screen
	entity-instance-create-type		Choose from the following values. At Start-Up At First Access As Required	Entity Bean instance generation mode	O	
messa-driven-eb?	jms	subscription-name?	Any character string	Subscriber identifier	O	Message-driven Bean Extended Information
		connection-factory-name?	Any character string	ConnectionFactory name	O	
		bean-pool-size?	Choose from the following values. 1−10000000	Number of initial start instances	O	
		destination-name?	Any character string	Destination name	O	
		retry-count?	Choose from the following values. 1−2147483647	Retry count	O	
		backup-connection-factory-name?	Any character string	JMS ConnectionFactory name for use against error loop	O	
		backup-destination-name?	Any character string	Destination name for use against error loop	O	
runas-entry	userid		Any character string	User ID	O	Security Identity
	password		Any character string	Password	O	

Tab Name			Value	Meaning	Editing	Correspondence to Interstage Management Console Screen
fujitsu-cmp-definition	datasource-name?		Any character string	Datasource name	O	CMF Mapping Definition
	schema-name?		Any character string	Schema name	O	
	table-name?		Any character string	Table name	O	
	select-for-update?		Choose from the following values. False True	Addition of FOR UPDATE clause to findByPrimaryKey method	O	finder Method Definition
	field-map+	field-name	Any character string When the correspondence of env-entry-name is not taken in export between the definition file and Enterprise Bean definition information, an error occurs.	Field name	X	CMF Mapping Definition
		field-type?	Any character string	Type	X	
		dbcolumn-name	Any character string	DB Column name	O	
	finder-map+	finder-key-name	Any character string When the correspondence of env-entry-name is not taken in export between the definition file and Enterprise Bean definition information, an error occurs.	Method signature	X	finder Method Definition
		finder-query-string	Any character string	Query	O	

Tab Name						Value	Meaning	Editing	Correspondence to Interstage Management Console Screen
fujitsu-cmp2x-mapping-definition (Specified exclusively from fujitsu-cmp-definition)	datasource-name?					Any character string	Data source name	O	CMF Mapping Definition
	ejb?	ejb-name				Any character string	Enterprise Bean name	O	
		schema-name				Any character string	schema-name	O	
		table-name				Any character string	table-name	O	
		field-map2x	field-map-entry2x*	default-dbcolumn-name		Any character string	CMP field name	O	
				is-primary-key		Choose from the following values. True False	Primary key	O	
				foreign-key?	foreign-ejb-name	Any character string	External key	O	CMR Mapping Definition
				field-name		Any character string	CMP field name	O	
				dbcolumn-name		Any character string	DB column name	O	
	join-object*	join-name				Any character string	Join table name	O	
		schema-name				Any character string	schema-name	O	
		table-name				Any character string	table-name	O	
		source	ejb-name			Any character string	Enterprise Bean name	O	
			cmr-field?			Any character string	CMR field name	O	
		sink	ejb-name			Any character string	Enterprise Bean name	O	
			cmr-field			Any character string	CMR field name	O	
		field-map2x				See ejb tag field-map2x		O	

Note

Note 1) This tag is invalid if it is defined for V7 or later.

Note 2) This tag is normally not used.

Note 3) A similar definition can be made when a function that is compatible with a previous version is used. Refer to the previous version manual for details of the form and the XML value.

- Tags with ? may be omitted.
- Tags with + may be repeatedly specified.
- If editing-enabled items with optional values are checked to see whether the specified value is included in the options. Items of the numeric value type are checked to see whether the value type and range are correct.
- If editing-enabled items exist in the Enterprise Bean definition information, they are changed to Enterprise Bean definition file values.
- If editing-enabled items are not included in the Enterprise Bean definition information, they are added to that information.
- The definition file value and Enterprise Bean definition information value of an editing-disabled item are compared. If these values differ, an error occurs.

Note

When the Enterprise Bean definition information is exported, the "password" of the Enterprise Bean definition information file displays "*" (one per password character). If the information is imported without changing this value, the character string is passed as is to the Enterprise Bean definition information.

DB Definition File Contents

This describes DB Definition File Contents the in XML format.

The file contents are as follows.

Table Details

Tab name: XML tab name in definition file

Value: XML value in definition file

Meaning: Tab meaning (the Customize Tool displays its item names.)

Editing: Whether values can be edited (O: Yes; X: No)

Correspondence to customize Tool screen: Correspondence to customize Tool screen

Table 13-4 DB Tab Value, Meaning and Correspondence

Tab name	Value	Meaning	Editing	Correspondence to Customize Tool screen
registration-name	Any character string	The JNDI registration name of the data source must be unique.	O	DB Access Environment Definition
driver-name?	Any character string	Driver class name	O	DB Access Environment Definition

Tab name	Value	Meaning	Editing	Correspondence to Customize Tool screen
driver-type	Choose from the following values. JDBC1.22 JDBC2.0	JDBC Driver version	O	DB Access Environment Definition
datasource-name?	Any character string	Datasource name	O	DB Access Environment Definition
url?	Any character string	URL of JDBC1.22 driver	O	DB Access Environment Definition
userid?	Any character string	User ID	O	DB Access Environment Definition
password?	Any character string	Password	O	DB Access Environment Definition
initial-context-factory ?	Any character string	Class name for JNDI service provider to access Naming Service	O	DB Access Environment Definition
provider-url?	Any character string	Class name for JNDI service provider to access Naming Service	O	DB Access Environment Definition

- The one to select the value checks whether the specified value exists in the selection leg.
- <registration-name> cannot set the value which overlaps with other <registration-name>.
- The item to which setting is mandatory has a different value specified by <driver-type>.
- Tags with ? may be omitted.
- Tags with + may be repeatedly specified.

The item which the DB definition file sets is different according to the version of JDBC used.

Table 13-5 Omissible Settings

Omissible	JDBC1.22	JDBC2.0
registration-name	O	O
driver-name	O	X
driver-type	O	O
datasource-name	X	O
url	O	X

Omissible	JDBC1.22	JDBC2.0
userid	O	Required when userid is set
password	O	Required when userid is set Required when userid is set
initial-context-factory	X	O
provider-url	X	O

O: Setting is required X: Setting is not required.

Note

When DB definition information is exported, the "password" of the DB definition file displays "*" (one per password character). If the information is imported without changing this value, the character string is passed as is to the DB definition information.

Enterprise Bean Definition File Example

This sample provides an Enterprise Bean definition file example.

```
<?xml version="1.0" encoding="Shift_JIS"?>
<!DOCTYPE ejbdef SYSTEM "ejbdef.dtd">
<ejbdef>
    <ejb-jar>
        <enterprise-beans>
            <session>
                <ejb-name>SampleCMPSession</ejb-name>

                <home>packageCMPSession.SampleCMPSessionHome</home>

                <remote>packageCMPSession.SampleCMPSessionRemote</remote>

                <ejb-class>packageCMPSession.SampleCMPSession</ejb-class>

                <session-type>Stateless</session-type>

                <transaction-type>Bean</transaction-type>

                <env-entry>
                    <env-entry-name>SampleCMPSession/Trace</env-entry-name>

                    <env-entry-type>java.lang.String</env-entry-type>

                    <env-entry-value>OFF</env-entry-value>
                </env-entry>

                <ejb-ref>
                    <ejb-ref-name>ejb/SampleCMP</ejb-ref-name>

                    <ejb-ref-type>Entity</ejb-ref-type>
```

```
                    <home>SampleCMPHome</home>

                    <remote>SampleCMPRemote</remote>

                    <ejb-link>SampleCMP</ejb-link>
                </ejb-ref>
            </session>
        </enterprise-beans>
    </ejb-jar>

    <fujitsu-bean-definition>
        <base>
            <jndi-name>SampleCMPSession</jndi-name>

            <tran-timeout>0</tran-timeout>

            <local-mode>False</local-mode>
        </base>

        <session-eb>
            <max-instance>16</max-instance>

            <session-timeout>0</session-timeout>

            <session-idle-timeout>600</session-idle-timeout>

            <max-ejbobject>1024</max-ejbobject>

            <stateless-instance-create-type>At First
Access</stateless-instance-create-type>

        </session-eb>
    </fujitsu-bean-definition>
</ejbdef>
```

DB Definition File Example

This sample provides a DB definition file example.

```
<?xml version="1.0" encoding="Shift_JIS"?>
<!DOCTYPE fujitsu-datasource-definition SYSTEM "ejbdbdef.dtd">
<fujitsu-datasource-definition>
    <datasource-entry>
        <registration-name>DS0001</registration-name>

        <driver-name>oracle.jdbc.driver.OracleDriver</driver-name>

        <driver-type>JDBC 1.22</driver-type>

        <url>jdbc:oracle:thin:@host:1521:TEST_DB</url>

        <userid>user</userid>

        <password>********</password>
    </datasource-entry>

    <datasource-entry>
        <registration-name>DS00011</registration-name>

        <driver-type>JDBC 2.0</driver-type>

        <datasource-name>jdbc/DataSourceName1</datasource-name>


<initial-context-factory><initial-context-factory>com.sun.jndi.fscontext.Ref
FSContextFactory</initial-context-factory>

        <provider-url>file:///c:/tmp/JNDI</provider-url>
    </datasource-entry>
</fujitsu-datasource-definition>
```

Chapter 14

Using the Interstage JDBC Driver

This chapter explains the Interstage JDBC Driver used to connect EJB applications to the SQL Server.

Microsoft(R) JDBC driver is recommended, although the Interstage JDBC driver is offered in Interstage Application Server. Refer to “Environment set up when SQL Server is used” before using Microsoft(R) JDBC driver to connect EJB applications to SQL Server

Overview of Interstage JDBC Driver

The Interstage JDBC Driver provides a function that enables an EJB application to link with an SQL Server via the JDBC interface.

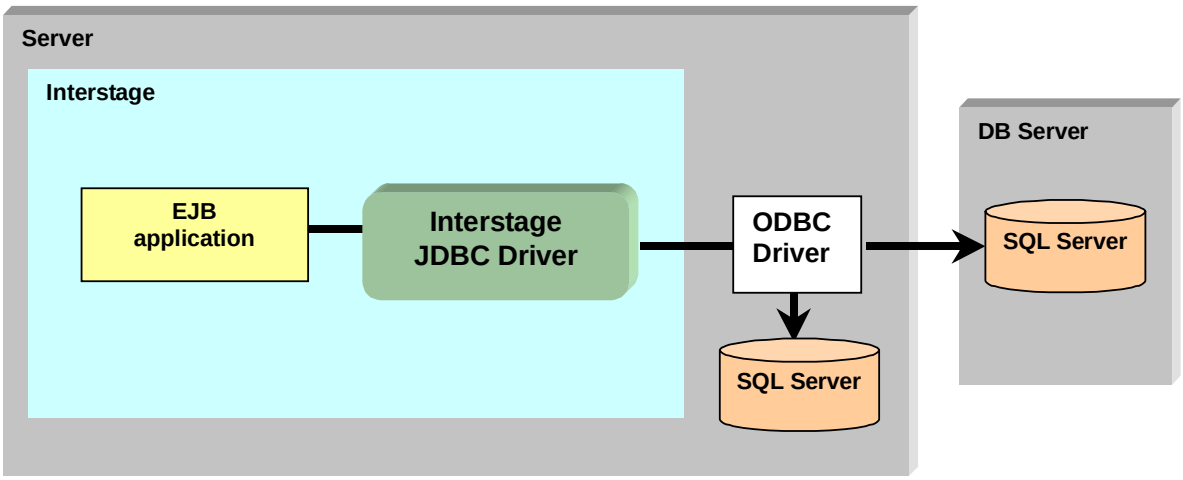


Figure 14-1 Linking Interstage with an SQL Server

Environment Setup Required for Connection to SQL Server

When SQL Server is used, the environment is set according to the following procedures.

1. Confirm the Java environment.
2. Set the environment variables

   Set the following environment variables. Set these environment variables to the WorkUnit definition with the Interstage Management Console when you use the WorkUnit. There is no need to set them to the WorkUnit definition for system environment variables.

Environment variables	Setting value
PATH	C:\Interstage\EJB\jdbc\bin
CLASSPATH	C:\Interstage\EJB\jdbc\lib\fjisjdbc2.jar

3. Register ODBC datasource

   The ODBC datasource administrator registers the ODBC datasource. Refer to “SQL Server Books Online” before registering the ODBC datasource with Windows NT (R).

Note

- Use system DSN for the ODBC datasource.
- Do not use the ODBC driver's Connection Pooling function.

4. Start JDBC Naming Service.

   Start the execution of the following commands and Naming Service.

   java com.fujitsu.interstage.jdbc.FJJdbcNameService [<port_no>]

   - [] This can be omitted.
   - < port_no>: The port number is specified here. This is 10526 by default.

5. Register the datasource with the JDBC datasource registration tool.

   Refer to Help of the JDBC datasource registration tool before registering the JDBC datasource.

6. Define the resource access.

   Use the Interstage Management Console to define the resource access. Refer to Help for the Interstage Management Console for details.

Methods of Connection to an SQL Server

Two methods are available for an EJB application to connect to an SQL Server using the Interstage JDBC Driver:

1. Using the Enterprise Bean Environment

   This method involves performing a lookup in the EJB application (as for the datasource defined by DB Access Environment Definition).

2. Using the Interstage JDBC Driver directly

   This method enables the use of the Interstage JDBC Driver directly from an EJB application.

Using the Enterprise Bean Environment

The following operations are required for connection to the SQL Sever using the Enterprise Bean Environment:

- Define a datasource in the DB access environment definition.
- Describe datasource lookup processing in an EJB application.

For the datasource definition, refer to J2EE Management Tool.

For writing datasource lookup processing in the EJB application, refer to Common.

Using the Interstage JDBC Driver Directly

When connecting to the SQL Server from the EJB application using the Interstage JDBC Driver directly, two connection methods are available. Describe either of the following operations in the EJB application.

- Operation after datasource connection
- Operation after URL connection

Datasource Connection Processing

For connection using the JDBC datasource, set the following parameters in the application.

Sample coding

```
Hashtable      env = new Hashtable();
env.put(Context.INITIAL_CONTEXT_FACTORY,<factory_class_name>);
  env.put(Context.PROVIDER_URL,<provider_url>);
  InitialContext ctx = new InitialContext(env);
  Datasource     ds  = (Datasource)ctx.lookup("jdbc/<datasource_name>");
  Connection     con = ds.getConnection();
```

<factory_class_name>

Specify com.fujitsu.interstage.jdbc.FJJdbcContextFactory.

This parameter cannot be omitted.

<provider_url>

Specify the URL used for connection to the naming service.

This parameter cannot be omitted.

The syntax is as follows:

```
FJIS://<host>[:<port_no>]
```

(the parameter inside [ ] is optional)

- **<host>**

  Specify the local host.

- **<port_no>**

  Specify the port number used for connection to the naming service. The default is 10526.

Example

The port number used for connection to the naming service is 10527.

```
FJIS://localhost:10527
```

<datasource_name>

Specify the JDBC datasource name that was specified when the JDBC datasource was stored.

This parameter cannot be omitted.

Example

The datasource name is MYDS.

```
Datasource ds = (Datasource)ctx.lookup("jdbc/MYDS")
```

URL Connection Processing

For connection using URL, set the following in the application program:

Sample coding

```
    Class.forName(<driver_name>);
    Connection con = DriverManager.getConnection(<url>,<user>,<password>);
```

<driver_name>

Specify com.fujitsu.interstage.jdbc.FJDriver.

This parameter cannot be omitted.

<url>

Specify the database to be connected to.

This parameter cannot be omitted.

The syntax is as follows:

```
jdbc:fjis:///<ODBC_DatasourceName>
```

- **<ODBC_DatasourceName>**

  Specify the datasource name that was set by the ODBC datasource administrator.

  This parameter cannot be omitted.

<user>

Specify the user ID used for connection to the database.

This parameter cannot be omitted.

<password>

Specify the password corresponding to the user ID used for connection to the database.

This parameter cannot be omitted.

Part IV

JTS/JTA Edition

Chapter 15

Using Java Transaction API (JTA)

JTA is a distributed transaction operation API supplied by a transaction service. The Java transaction service (JTS) is the transaction service supplied by Interstage.

JTA

This section explains the following topics:

- JTA Interfaces
- User Transaction Interface

JTA Interfaces

The JTA specifications were proposed as a standard Java interface by Sun Microsystems, Inc.

Components embedded in a distributed transaction are J2EE application and resource manager.

JTA enables multiple components to be interlinked so that they can be handled as one transaction.

JTA has the following interfaces.

Table 15-1 JTA Interfaces

Interface	Description
javax.transaction.UserTransaction	Interface that instructs the start or completion of a transaction. This interface can be used by an application.
javax.transaction.TransactionManager	Interface used by the transaction manager. This interface cannot ordinarily be used from an application.
javax.transaction.Transaction	Interface that logically handles a transaction. Usually, this interface cannot be used by an application.
javax.transaction.Synchronization	Interface that performs completion and synchronous processing of a transaction. Usually, this interface cannot be used by an application.
javax.transaction.Status	Interface that defines the state of a transaction. The value returned in the getStatus method of UserTransaction is defined in this interface.
javax.transaction.xa.XAResource	Interface between the transaction manager and resource manager. Usually, this interface need not be used by an application.
javax.transaction.xa.Xid	Identifier interface used when the resource manager handles a transaction. Usually, this interface need not be used by an application.

Note

A JTA that can be used by applications is the UserTransaction interface. The TransactionManager interface cannot be used in an application.

User Transaction Interface

The User Transaction interface is included inJTA.

The user transaction interface has the functions listed in Table 15-2.

Table 15-2 Functions Provided by the User Transaction Interface

Function name	Description
begin	Starts a transaction, and associates the thread with the transaction.
commit	Completes a commit operation on the transaction corresponding to the thread.
getStatus	Obtains the status of the transaction related to the thread.
rollback	Completes a rollback operation on the transaction related to the thread.
setRollbackonly	Allows only rollback to be performed on the corresponding transaction.
setTransactionTimeout	Sets the transaction timeout. A transaction timeout must be specified before a transaction starts.

Note

A transaction timeout must be specified using the setTransactionTimeout method before a transaction starts.

If a transaction timeout is specified after a transaction starts, it is not applied to the started transaction.

Environment Setup for the User Transaction Interface

To use the User Transaction interface in a J2EE application other than an EJB application, define the following class libraries in a class path.

- fjtsclient.jar (class library for JTS clients)
- isj2ee.jar (class library of J2EE)
- class library of CORBA service
- class library for EJB service (required for creating an EJB client application)

Note

Class libraries for JTS client can be installed only by the server function. They are not installed by the client function. To run a JTA application in environment in which the client function is installed, class libraries for JTS client must be copied from the environment in which the server function is installed.

Class libraries are stored in the following locations on the environment in which the server function is installed.

Class library for clients

Storage locations

Windows

C:\INTERSTAGE\ots\lib\fjtsclient.jar

Solaris OE

/opt/FSUNots/lib/fjtsclient.jar

Linux

/opt/FJSVots/lib/fjtsclient.jar

Note

When using the User Transaction interface in an EJB application, do not define a JTS class library in the environment variable class path.

Acquiring the User Transaction Interface

This section explains how to acquire the UserTransaction object from JNDI to use the User Transaction interface.

To acquire the UserTransaction object from JNDI, define JNDI environment properties.

For details on the JNDI environment properties, see Setting Details of JNDI Environment Properties in Chapter 6 .

When acquiring the UserTransaction object from JNDI, specify the JNDI name below.

```
java:comp/UserTransaction
```

Example

```
InitialContext ic = new InitialContext();
UserTransaction ut = ic.lookup("java:comp/UserTransaction");
```

Note

If JNDI environment properties are not correctly defined, lookup processing fails. In this case, confirm the setting of the JNDI environment properties.

Generating a JTA Application

This section explains how to generate a JTA application.

Application Configuration

Figure 15-1 shows an example of a client application configuration that takes the following processes into account.

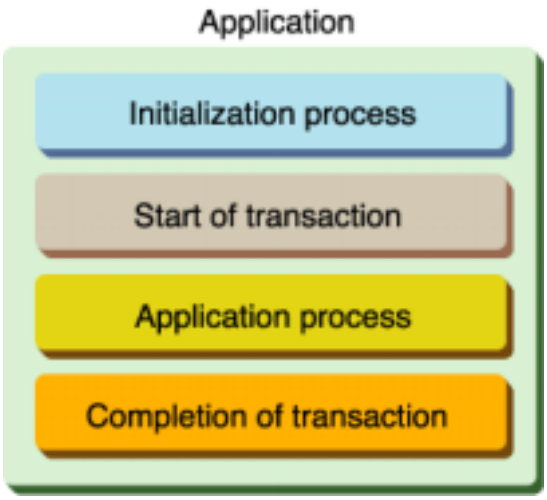


Figure 15-1 Client Application Configuration

Initialization process

Use JNDI to acquire the UserTansaction object.

Start a transaction

Start a transaction.

Application process section

Describe the business logic.

Completion of transaction

Commit or roll back a transaction.

Performing Initialization Process and Acquiring the UserTransaction object

Generate InitialContext, and then acquire the javax.transaction.UserTransaction object.

When acquiring the UserTransaction object from JNDI, specify the JNDI name below.

```
java:comp/UserTransaction
```

Example

The following is a process description example.

```
/* Initialization process*/
javax.transaction.UserTransaction ut = null;
javax.naming.Context ic = null;
// Create of InitialContext
try{
    ic = new InitialContext();
} catch( NamingException e ) {
    System.out.println( "error: new InitialContext()" );
    System.exit(1);
}
// Acquisition of UserTransaction
try {
    ut = (UserTransaction)ic.lookup(“java:comp/UserTransaction”);
} catch( NamingException e ) {
    System.out.println( "error: lookup UserTransaction" );
    System.exit(1);
}
```

From Transaction Start to Transaction Stop

To start a transaction with the UserTransaction interface, issue the begin method.

To complete a transaction with the UserTransaction interface, issue the commit or rollback method. The commit method is used to determine a kind of processing and the rollback method is used to cancel the processing.

Example

An example of description of processing is as follows.

```
// Acquisition of UserTransaction
try {
    ut = (UserTransaction)ic.lookup(“java:comp/UserTransaction”);
} catch( NamingException e ) {
    System.err.println( "error: lookup UserTransaction" );
    System.exit(1);
}
try {
    ut.begin();
} catch (NotSupportedException e) {
```

```
    System.err.println("The thread is already associated with a transaction
");
    System.exit(1);
} catch(SystemException e) {
    System.err.println("The system error has been encountered");
    System.exit(1);
}

try {
// Describe the business logic.
} catch(Throwable e) {
// In this example, when the process fails, the flag is set to false.
commitRequest = false;
}

if(commitRequest) {
    try {
        ut.commit();
    } catch(RollbackException e) {
        System.err.println("The transaction has been rolled back rather than
committed");
    } catch(SystemException e) {
        System.err.println("The system error has been encountered.");
    }
} else {
    try {
        ut.rollback();
    } catch(java.lang.IllegalStateException e) {
        System.err.println("The current thread is not associated with a
transaction.");
    } catch(SystemException e) {
        System.err.println("The system error has been encountered.");
    }
}
```

JTA Application Example

```
/* Initialization process*/
javax.transaction.UserTransaction ut = null;
javax.naming.Context ic = null;
// Create of InitialContext
try{
    ic = new InitialContext();   (1)
} catch( NamingException e ) {
    System.out.println( "error: new InitialContext()" );
    System.exit(1);
}
// Acquisition of UserTransaction
try {
    ut = (UserTransaction)ic.lookup(“java:comp/UserTransaction”);   (2)
} catch( NamingException e ) {
    System.err.println( "error: lookup UserTransaction" );
    System.exit(1);
```

```
}
try {
    ut.begin();   (3)
} catch (NotSupportedException e) {
    System.err.println("The thread is already associated with a transaction
");
    System.exit(1);
} catch(SystemException e) {
    System.err.println("The system error has been encountered");
    System.exit(1);
}

try {
    // Describe the business logic.     (4)
} catch(Throwable e) {
    // In this example, when the process fails, the flag is set to false.
    commitRequest = false;
}

if(commitRequest) {   (5)
    try {
        ut.commit();   (6)
    } catch(RollbackException e) {
        System.err.println("The transaction has been rolled back rather than
committed");
    } catch(SystemException e) {
        System.err.println("The system error has been encountered.");
    }
} else {
    try {
        ut.rollback();   (6)
    } catch(java.lang.IllegalStateException e) {
        System.err.println("The current thread is not associated with a
transaction.");
    } catch(SystemException e) {
        System.err.println("The system error has been encountered.");
    }
}
```

1. To use JNDI, generate InitialContext.
2. Acquire the javax.transaction.UserTransaction object from JNDI.
3. Start a transaction using the javax.tansaction.UserTransaction.begin method.
4. Describe a business process.
5. Confirm whether the transaction can be committed, and determine the state of the transaction.
6. When determining and terminating the transaction, use the commit method to commit the transaction.
7. When not committing the transaction because of an error, use the rollback method to roll back the transaction.

Note

For details on JNDI, refer to JNDI in Chapter 4.

Precautions

When the Client Application Failed to be Activated

Confirm whether necessary information is defined in environment variables.

- fjtsclient.jar (class library for JTS clients)
- isj2ee.jar (class library of J2EE)
- class library of CORBA service
- class library for EJB service (required for creating an EJB client application)

For environment properties necessary for JNDI, refer to JNDI in Chapter 4.

If the Client Application Detects an Error

If the client application detects an error as detailed below, terminate the transaction by issuing the rollback command:

- If the client application detects an abnormality within its own programs.
- If the server application notifies the client application of an error.

Part V

JMS Edition

Chapter 16

Environment Settings for Interstage JMS

This chapter describes the environment settings for using Interstage JMS.

Environment Settings for the Event Channel Operation Machine

This section describes the environment settings for the machine that performs operation of the event channel used for sending/receiving messages by the JMS application.

The following table lists the commands provided by the Event Service.

Table 16-1 List of Commands of the Event Service

Classification	Command name	Outline
Environment	essetcnf	Administers Event Service configuration information
Event Service	esmonitor	Displays Event Service status
Event Factory	esstartfctry	Starts an Event Channel factory
	esstopfctry	Stops an Event Channel factory
Unit	esmkunit	Generates an Unit
	esrmunit	Deletes an Unit
	esstartunit	Starts an Unit
	esstopunit	Stops an Unit
Event Channel	esmkchnl	Generates an Event Channel
	esrmchnl	Deletes an Event Channel
	esstartchnl	Starts an Event Channel
	esstopchnl	Stops an Event Channel
	essetcnfchnl	Sets an operating environment for Event Channels
	esmonitorchnl	Displays Event Channel connection information

The following table shows the correspondence between terms used in the Event Service and terms used in Interstage JMS.

Table 16-2 List of Terms used in the Event Service and Interstage JMS

Event Service	Interstage JMS
Event Channel	Topic or Queue
Statically generated channels	Topic or Queue
Dynamically generated channels	TemporaryTopic or TemporaryQueue
Consumer	Subscriber or Receiver
Supplier	Publisher or Sender
Event Data	Message

Environment Setting before Operation

The following figure shows the procedure for environment setting before the operation.

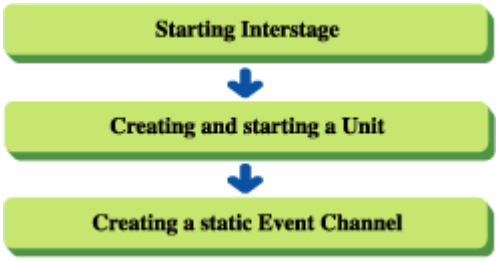


Figure 16-1 Event Channel Operation Machine Environment Setting Operations

Starting Interstage

Activate the Event Service by starting Interstage with the *isstart* command.

```
isstart
```

Note

After adding the line 'Event Service=yes' to the Interstage operating environment definition file, initialize Interstage using the *isinit* command.

Creating and Starting a Unit

To use the following functions, create a unit with the *esmkunit* command.

- Durable Subscription function
- Event channel unvolatilizing function (message assurance function)
- Local transaction function (message assurance function)
- Global transaction function (message assurance function)

Example

Creating a unit using the unit definition file 'unit1.def'.

```
esmkunit -uf unit1.def
```

Edit the following items in the unit definition file as required:

- **unitmode**

  Specify std (standard unit) or ext (extended unit). To use the global transaction function, specify ext.

- **trandir**

  Specify the directory to store the transaction files. For Windows®, the directory must be in NTFS.

- **sysdir**

  Specify the directory to store the system files. For Windows®, the directory must be in NTFS.

- **userdir**

  Specify the directory to store event data files. For Windows®, the directory must be in NTFS.

The 'esunit01.def' prototype of a unit definition file is in the following directory (default installation path):

Windows

```
C:\Interstage\eswin\etc\def
```

Solaris OE Linux

```
/opt/FJSVes/etc/def
```

Then, start the unit by using the *esstartunit* command.

Example

Starting the unit name 'unit1'

```
esstartunit -unit unit1
```

Creating a Static Event Channel

Create an event channel used for sending/receiving messages by the JMS application by using the *esmkchnl* command.

Specify the following options and arguments as required:

- **-notify**

  This option is required.

- **-ptp**

  For the Point-To-Point messaging model, specify this option. When this option is omitted, the Publish/Subscribe messaging model is assumed.

- **-persist all**

  Specify -persist all when the Durable Subscription function, persistent function of the event channel, local transaction function, or global transaction function is used.

- **-tran**

  Specify -tran when the global transaction function is not used.

- **-ots**

  Specify -ots when the global transaction function is used.

Example

For Publish/Subscribe messaging model:

Specify as shown below when the event channel 'mychannel' for operation using the persistent function of the event channel and local transaction function should be created in the group 'mygroup':

```
esmkchnl -g mygroup -c mychannel -notify -persist all -tran
```

For Point-To-Point messaging model:

Specify as shown below when the event channel 'mychannel' for operation using the persistent function of the event channel and local transaction function should be created in the group 'mygroup':

```
esmkchnl -g mygroup -c mychannel -notify -ptp -persist all -tran
```

Note

Check that the database linkage service has been started before global transaction operation.

Ensure you specify a database linkage service concurrency value higher than the total number of subscribers and publishers (or receivers and senders).

For more information, refer to the JTS Operation chapter.

In the Point-To-Point messaging model, change the timeout value of the local transaction (default: 604,800 seconds) to a smaller value (about 300 seconds is recommended) with the *essetcnfchnl* command (-ltrntime option specification).

Changing the Event Channel Operating Environment

The event channel operating environment can be changed by:

- Setting the event service configuration information with the essetcnf command
- Setting the event channel environment information with the essetcnfchnl command

The following describes the command options.

Table 16-3 *essetcnf* Command and *essetcnfchnl* Command Options

Option	Description
-schmax	Maximum number of activations of the static event channel used by a topic or queue
-dchmax	Maximum number of activations of the dynamic event channel used by a temporary topic or temporary queue
-edinit	Initial number of messages that can be stored in an event channel
-edmax	Maximum number of messages that can be stored in an event channel
-wtime	Message waiting time (second) (*1)
-coninit	Initial number of subscribers or receivers that can be connected to an event channel
-conext	Extended number of subscribers or receivers that can be connected to an event channel

Option	Description
-conenum	Number of extensions of subscribers or receivers that can be connected to an event channel
-supinit	Initial number of publishers or senders that can be connected to an event channel
-supext	Extended number of publishers or senders that can be connected to an event channel
-supenum	Number of extensions of publishers or senders that can be connected to an event channel
-logsize	Size of the log file (in Kbytes) which stores error information from the Event Service
-loglevel	Error information level sent to the log file
-logdump	Event data dump size sent to the log file
-gtrnmax	Number of global transactions that can be executed simultaneously
-ltrntime	Local transaction timeout period (in seconds)
-2pctime	Two-phase commit timeout monitoring period (in seconds)
-retrytime	Retry interval during recovery (in seconds)
-retrymax	Recovery retry count
-chkcon	Error return mode during disconnection of subscribers
-dcache	Number of memory caches for messages at persistent operation of an event channel

*1) A value of ten seconds or more is recommended as the waiting time of the event data. If setting a wait time of less than 10 seconds, configure operations that do not wait for event data using the receiveNoWait() method.

This value varies with the period_receive_timeout value of the CORBA service operating environment file (config). The timeout for receiving messages specified by receive() is checked at the intervals set for the wait time. Therefore, the timeout for receiving messages is returned as follows:

- For 'Wait time for event data >= Value of receive()'

  The timeout for receiving messages is returned as 'Wait time for event data'.

  Example

  Wait time for event data: '40 seconds', Value of receive() '10': 40 seconds

- For 'Wait time for event data < Value of receive()'

  The timeout for receiving messages is returned as 'Wait time for event data * Number of checks'.

  Example

  Wait time for event data: '40 seconds', Value of receive() '50': 80 (40 * 2) seconds

Note

When the environment of an Event Service in persistent operation is modified, only the following configuration information and environment information can be modified.

- -wtime
- -dtime
- -logsize
- -loglevel
- -logdump
- -ltrntime
- -2pctime
- -chkcon

If you try to change other information, the operating environment of the event channel in persistent operation may be modified. This means the integrity of the persistent information may not be retained, and the event channel in persistent operation must be recreated.

Environment Deletion after Operation

The following figure shows the procedure for environment deletion after the operation:

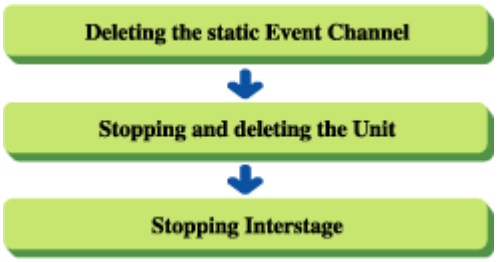


Figure 16-2 Event Channel Operation Machine Environment Deletion Operations

Deleting the Static Event Channel

Delete the event channel used for sending/receiving messages by the JMS application by using the *esrmchnl* command.

Example

Specify as shown below to delete all event channels belonging to the group 'mygroup'.

```
esrmchnl -g mygroup
```

Note

If a unit is being used, you must activate it by using the *esstartunit* command.

Stopping and Deleting a Unit

If a unit is being used, forcibly stop it by using the *esstopun*it command.

Example

Stopping the unit name 'unit1' forcibly.

```
esstopunit –unit unit1 -o off
```

Then, delete the unit by using the *esrmunit* command.

Example

Deleting the unit name 'unit1'.

```
esrmunit –unit unit1
```

Stopping Interstage

Stop the Event Service forcibly by stopping Interstage using the *isstop* command.

```
isstop -f
```

Environment Settings for the JMS Application Operation Machine

This section describes the environment settings for the machine that operates JMS applications.

The following definitions are needed for operation of JMS applications by the user:

- JNDI environment definition

  Definition needed for the JMS application to access the Naming Service of JNDI

- ConnectionFactory definition

  Definition needed to connect to the Interstage JMS provider

- Destination definition

  Definition information of the destination to/from which messages are sent/received by the JMS application

The following table lists the commands provided by Interstage JMS:

Table 16-4 List of Commands of Interstage JMS

Classification	Command name	Outline
ConnectionFactory	jmsmkfact	Registers ConnectionFactory definition
	jmsrmfact	Deletes ConnectionFactory definition
	jmsinfofact	Lists ConnectionFactory definitions
Destination	jmsmkdst	Registers Destination definition
	jmsrmdst	Deletes Destination definition
	jmsinfodst	Lists Destination definitions
durable Subscriber	jmsrmds	Deletes durable Subscriber
	jmsinfods	Lists durable Subscribers

Check that the paths and class files required for the operation of JMS applications are defined in the following environment variables (default installation path).

Windows

- **Environment variable PATH**

  JDK path (Note 1)

  C:\Interstage\J2EE\bin (Note 2)

  C:\Interstage\jms\bin (Note 2)

- **Environment variable CLASSPATH**

  C:\Interstage\ODWIN\etc\Class\ODjava2.jar (Note 3)

  C:\Interstage\eswin\lib\esnotifyjava2.jar (Note 4) (Note 5)

  C:\Interstage\J2EE\lib\isj2ee.jar

  C:\Interstage\jms\lib\fjmsprovider.jar

  C:\Interstage\ots\lib\fjtsclient.jar (Note 6) (Note 7)

Solaris OE

- **Environment variable PATH**

  JDK path (Note 1)

  /opt/FJSVj2ee/bin

  /opt/FJSVjms/bin

- **Environment variable CLASSPATH**

  /opt/FSUNod/etc/class/ODjava2.jar (Note 3)

  /opt/FJSVes/lib/esnotifyjava2.jar (Note 5)

  /opt/FJSVj2ee/lib/isj2ee.jar

  /opt/FJSVjms/lib/fjmsprovider.jar

  /opt/FSUNots/lib/fjtsclient.jar (Note 6)

- **Environment variable LD_LIBRARY_PATH**

  /opt/FSUNod/lib

  /opt/FJSVjms/lib

Linux

- **Environment variable PATH**

  JDK path (Note 1)

  /opt/FJSVj2ee/bin

  /opt/FJSVjms/bin

- **Environment variable CLASSPATH**

  /opt/FJSVod/etc/class/ODjava2.jar (Note 3)

  /opt/FJSVes/lib/esnotifyjava2.jar (Note 5)

  /opt/FJSVj2ee/lib/isj2ee.jar

  /opt/FJSVjms/lib/fjmsprovider.jar

  /opt/FJSVots/lib/fjtsclient.jar (Note 6)

- **Environment variable LD_LIBRARY_PATH**

  /opt/FJSVod/lib

  /opt/FJSVjms/lib

Note 1) If multiple JDKs are installed, make a setting so that the JDK to be used has a valid path.

Note 2) On Windows® 9x and Windows® Me, you need to set the PATH variable as a system environment variable after installation.

Note 3) If JDK 1.4 is used, set the following class file:

ODjava4.jar

Note 4) If the Interstage client function is installed, set the following class file:

C:\Interstage\ODWIN\etc\Class\esnotifyjava2.jar

Note 5) If the JDK 1.4 is used, set the following class file:

esnotifyjava4.jar

Note 6) Required to use the global transaction function.

Note 7) Must be retrieved by the database linkage service from the host in which the function is installed.

Note

Check that the CORBA service (ObjectDirector) is specified in the environment setup file as an ORB to be used.

Create a text file called orb.properties associated with the ORB to be used, then save this text file under 'lib' within the directory defined in the Java system properties file 'java.home.' Save according to the instructions below for each installed package, including: Interstage Java Server Package (for Linux, Java execution environment), Interstage Apworks Client Runtime Package, and the JDK/JRE.

Windows

Interstage Java Server Package

```
For JRE1.3
  <Installation directory of Interstage Java Server Package>\jbk3\jre\lib
For JDK1.3
  <Installation directory of Interstage Java Server Package>\jbk3\jdk\jre\lib
For JDK1.4
  <Installation directory of Interstage Java Server Package>\jbk4\jdk\jre\lib
```

Interstage Apworks Client Runtime Package

```
For JRE1.3
  <Installation directory of Interstage Apworks Client Runtime
Package>\jbk3\jre\lib
For JRE1.4
  <Installation directory of Interstage Apworks Client Runtime
Package>\jbk4\jre\lib
```

The function equivalent to Interstage Apworks Client Runtime Package (See Note below)

```
For JRE1.3
  <Installation directory of the function equivalent to Interstage Apworks Client
Runtime Package>\jre13\lib
For JRE1.4
  <Installation directory of the function equivalent to Interstage Apworks Client
Runtime Package>\jre14\lib
```

Note

Interstage Client function installation directory locations (equivalent to Interstage Apworks Client Runtime Package, included with the Interstage Java Server Package or JRE JBK plug-in).

Solaris OE Linux

Interstage Java Server Package (for Linux, Java execution environment)

```
For JRE1.3
  <Installation directory of Interstage Java Server Package>/jre13/lib
For JRE1.4
  <Installation directory of Interstage Java Server Package>/jre14/lib
For JDK1.3
  <Installation directory of Interstage Java Server Package>/jdk13/jre/lib
For JDK1.4
<Installation directory of Interstage Java Server Package>/jdk14/jre/lib
```

Example settings for the orb.properties file:

```
  org.omg.CORBA.ORBClass=com.fujitsu.ObjectDirector.CORBA.ORB
  org.omg.CORBA.ORBSingletonClass=com.fujitsu.ObjectDirector.CORBA.
SingletonORB
```

Environment Setting before Operation

The following figure shows the procedure for environment setting before operation:

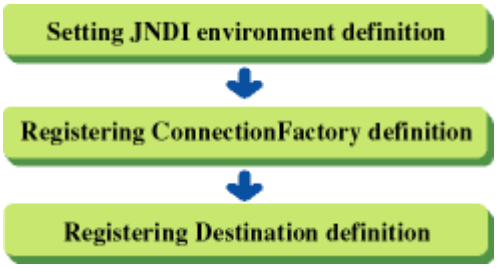


Figure 16-3 Application Operation Machine Environment Setting

Setting JNDI Environment Definitions

To allow a JMS application to access the JNDI naming service, you need to specify an environment property or specify a resource manager name in the reference resource information of deployment descriptor.

- For information on a client container, refer to Setting J2EE Application Clients in Chapter 6.
- For information on an EJB container, refer to Common in Chapter 28.
- For information on a Web container, refer to The Servlet Service Environment Definition Files in Chapter 11.

Registering ConnectionFactory Definition

Register the ConnectionFactory definitions to be referenced by the JMS application by using the *jmsmkfact* command.

Specify the following option or argument as required:

- **-t**

  Specify when ConnectionFactory is of TopicConnectionFactory type.

- **-q**

  Specify when ConnectionFactory is of QueueConnectionFactory type.

- **-x**

  Specify -x to use the global transaction function.

Example

For Publish/Subscribe messaging model:

Specify as shown below to register the ConnectionFactory definition of TopicConnectionFactory type whose client ID is 'client' and JNDI name is 'java:comp/env/jms/TestTopicConnectionFactory'.

```
jmsmkfact -t -i client TestTopicConnectionFactory
```

For Point-To-Point messaging model:

Specify as shown below to register the ConnectionFactory definition of QueueConnectionFactory type whose client ID is 'client' and JNDI name is 'java:comp/env/jms/TestQueueConnectionFactory'.

```
jmsmkfact -q -i client TestQueueConnectionFactory
```

Note

The ConnectionFactory definition can also be registered using the J2EE Management tool or the J2EE resource access definition.

Registering Destination Definition

Register the Destination definitions to be referenced by the JMS application by using the *jmsmkdst* command.

Specify the following option or argument as required:

- **-t**

  Specify when Destination is of Topic type.

- **-q**

  Specify when Destination is of Queue type.

Example

For Publish/Subscribe messaging model:

Specify as shown below to associate the event channel 'mychannel' of the group 'mygroup' with the Destination definition of Topic type whose JNDI name is 'java:comp/env/jms/TestTopic'.

```
jmsmkdst -t -g mygroup -c mychannel TestTopic
```

For Point-To-Point messaging model:

Specify as shown below to associate the event channel 'mychannel' of the group 'mygroup' with the Destination definition of Queue type whose JNDI name is 'java:comp/env/jms/TestQueue'.

```
jmsmkdst -q -g mygroup -c mychannel TestQueue
```

Note

The Destination definition can also be registered using the J2EE Management tool or the J2EE resource access definition.

Environment Setup during Web Application Operation

To use JMS in a Web application, you need to make the following settings:

JServlet environment definition file (jswatch.conf)

- Environment variables

  Set the following environment variables in [containername].env.

  – Environment variable PATH

  – Environment variable CLASSPATH

  – Environment variable LD_LIBRARY_PATH

Note

For information on the environment variables to be specified, refer to Environment Settings for the JMS Application Operation Machine in Chapter 39.

The following shows some setting examples.

Windows

```
# Path settings
 [containername].env=PATH=C:\Interstage\J2EE\bin
 [containername].env=PATH=C:\Interstage\jms\bin
# Class path settings
 [containername].env=CLASSPATH=C:\Interstage\ODWIN\etc\class\ODjava2.jar
 [containername].env=CLASSPATH=C:\Interstage\eswin\lib\esnotifyjava2.jar
 [containername].env=CLASSPATH=C:\Interstage\J2EE\lib\isj2ee.jar
 [containername].env=CLASSPATH=C:\Interstage\jms\lib\fjmsprovider.jar
```

Solaris OE

```
# Path settings
 [containername].env=PATH=/opt/FJSVj2ee/bin
 [containername].env=PATH=/opt/FJSVjms/bin
# Class path settings
 [containername].env=CLASSPATH=/opt/FSUNod/etc/class/ODjava2.jar
 [containername].env=CLASSPATH=/opt/FJSVes/lib/esnotifyjava2.jar
 [containername].env=CLASSPATH=/opt/FJSVj2ee/lib/isj2ee.jar
 [containername].env=CLASSPATH=/opt/FJSVjms/lib/fjmsprovider.jar
# Library path settings
 [containername].env=LD_LIBRARY_PATH=/opt/FSUNod/lib:/opt/FJSVjms/lib
```

Linux

```
# Path settings
 [containername].env=PATH=/opt/FJSVj2ee/bin
 [containername].env=PATH=/opt/FJSVjms/bin
# Class path settings
 [containername].env=CLASSPATH=/opt/FJSVod/etc/class/ODjava2.jar
 [containername].env=CLASSPATH=/opt/FJSVes/lib/esnotifyjava2.jar
 [containername].env=CLASSPATH=/opt/FJSVj2ee/lib/isj2ee.jar
 [containername].env=CLASSPATH=/opt/FJSVjms/lib/fjmsprovider.jar
# Library path settings
 [containername].env=LD_LIBRARY_PATH=/opt/FJSVod/lib:/opt/FJSVjms/lib
```

- Startup parameters

  Set an environment property in [containername].bin.parameters to enable the object reference function.

Web application environment definition file (web.xml)

Make a ConnectionFactory definition.

For information on a JServlet environment and a Web application environment definition file, refer to The Servlet Service Environment Definition Files in Chapter 11.

Environment Deletion after Operation

The following figure shows the procedure for environment deletion after operation:

Figure 16-4 Application Operation Machine Environment Deletion

Deleting ConnectionFactory Definition

Delete the ConnectionFactory definitions by using the *jmsrmfact* command.

Example

Specify as shown below to delete the ConnectionFactory definition whose JNDI name is 'java:comp/env/jms/TestTopicConnectionFactory'.

```
jmsrmfact TestTopicConnectionFactory
```

Note

The ConnectionFactory definition can also be deleted using the J2EE Management tool or the J2EE resource access definition.

Deleting Destination Definition

Delete the Destination definitions by using the *jmsrmds*t command.

Example

Specify as shown below to delete the Destination definition whose JNDI name is 'java:comp/env/jms/TestTopic'.

```
jmsrmdst TestTopic
```

Note

The Destination definition can also be deleted using the J2EE Management tool or the J2EE resource access definition.

Deleting Durable Subscriber

When the Durable Subscription function was used, if a durable Subscriber is not deleted by an application with the unsubscribe method, a durable Subscriber must be deleted with the *jmsrmdst* command.

Example

Specify as shown below to delete the durable Subscriber whose Durable Subscription name is 'dsub' and client ID is 'client'.

```
jmsrmds -n dsub -i client
```

Durable Subscription name and client ID can be checked with the *jmsinfods* command.

Note

If a durable Subscriber is deleted, you must activate the event channel by using the *esstartchnl* command.

Chapter 17

Developing a JMS Application

This chapter describes the development procedure of a JMS application.

Designing an Application

Design a JMS application according to its purpose as follows:

- When a message arrives at the destination, it should be processed automatically

  Use the Message Listener.

- In the Publish/Subscribe messaging model, messages delivered when the receiving JMS application is stopped should be received

  Use the Durable Subscription function.

- Message losses should be prevented

  Use the message persistent function and the local transaction function.

- Message processing and database processing should be guaranteed consistently

  Use the global transaction function.

- When a receiver application wants to receive only the interested information

  Use the message selector function.

- When a channel is not created for each sender site or receiver site to save resources

  Use the message selector function.

- When the environment of an event service does not need to be changed even if a sender site or receiver site is changed frequently

  Use the message selector function.

- In the Point-To-Point messaging model, when messages stored in the queue need to be monitored

  Use the queue browser function.

Creating a JMS Application

Create an application in accordance with the Java Message Service 1.0.2 specification released by US Sun Microsystems, Inc.

Publish/Subscribe Messaging Model

This section describes developing an application in the Publish/Subscribe messaging model.

In the Publish/Subscribe messaging model, a publisher (sending application) and a subscriber (receiving application) are used.

A publisher sends a message to the event channel while a subscriber requests the event channel for a message.

Creating a Publisher

A publisher sends a message to the event channel. The following provides a procedure example and a processing flow in which a publisher sends a message to the event channel.

Publisher

```
public class Publisher {
  public static void main() {
    ...
    try {
      InitialContext initialContext = new InitialContext();                /*
1 */
      TopicConnectionFactory topicConnectionFactory = (TopicConnectionFactory)
        initialContext.lookup("java:comp/env/jms/TopicConnectionFactory");
/* 2 */
      Topic topic = (Topic)initialContext.lookup("java:comp/env/jms/Topic');
/* 3 */
      TopicConnection topicConnection =
        topicConnectionFactory.createTopicConnection();                    /*
4 */
      TopicSession topicSession =
        topicConnection.createTopicSession(false, Session.AUTO_ACKNOWLEDGE);
/* 5 */
      TopicPublisher topicPublisher = topicSession.createPublisher(topic);
/* 6 */
      topicPublisher.publish(Message);                                     /*
7 */
      topicConnection.close();                                             /*
8 */
```

```
    } catch( Exception e ) {
      ...
    }
    ...
  }
}
```

1. Construct a JNDI start context.
2. Acquire a TopicConnectionFactory object (if the JNDI name is 'TopicConnectionFactory').
3. Acquire a Topic object (if the JNDI name is 'Topic').
4. Create TopicConnection.
5. Create TopicSession.
6. Create TopicPublisher.
7. Send a message.
8. Close TopicConnection.

Note

For information on the specification of an environment property when a JNDI start context is constructed, refer to 'Setting Details of JNDI Environment Properties' in Chapter 6.

Creating a Subscriber

A subscriber requests the event channel for a message. The following provides a procedure example and a processing flow in which a subscriber requests the event channel for a message.

Subscriber

```
public class SubscriberS {
  public static void main() {
    ...
    try {
      InitialContext initialContext = new InitialContext();                /*
1 */
      TopicConnectionFactory topicConnectionFactory = (TopicConnectionFactory)
        initialContext.lookup("java:comp/env/jms/TopicConnectionFactory");
/* 2 */
      Topic topic = (Topic)initialContext.lookup("java:comp/env/jms/Topic");
/* 3 */
      TopicConnection topicConnection =
        topicConnectionFactory.createTopicConnection();                    /*
4 */
      TopicSession topicSession =
        topicConnection.createTopicSession(false, Session.AUTO_ACKNOWLEDGE);
/* 5 */
      TopicSubscriber topicSubscriber = topicSession.createSubscriber(topic);
/* 6 */
      topicConnection.start();                                             /*
7 */
```

```
      Message message = topicSubscriber.receive();                    /*
8 */
      topicConnection.close();                                        /*
9 */
    } catch( Exception e ) {
      ...
    }
    ...
  }
}
```

1. Construct a JNDI start context.
2. Acquire a TopicConnectionFactory object (if the JNDI name is 'TopicConnectionFactory').
3. Acquire a Topic object (if the JNDI name is 'Topic').
4. Create TopicConnection.
5. Create TopicSession.
6. Create TopicSubscriber.
7. Start the delivery of a message upon connection.
8. Receive a message.
9. Close TopicConnection.

Note

For information on the specification of an environment property when a JNDI start context is constructed, refer to 'Setting Details of JNDI Environment Properties' in Chapter 6.

Point-To-Point Messaging Model

This section describes developing an application in the Point-To-Point messaging model.

In the Point-To-Point messaging model, a sender (sending application) and a receiver (receiving application) are used.

A sender sends a message to the event channel while a receiver requests the event channel for a message.

Creating a Sender

A sender sends a message to the event channel. The following provides a procedure example and a processing flow in which a sender sends a message to the event channel.

Sender

```
public class Sender {
  public static void main() {
    ...
    try {
      InitialContext initialContext = new InitialContext();                /*
1 */
      QueueConnectionFactory queueConnectionFactory = (QueueConnectionFactory)
        initialContext.lookup("java:comp/env/jms/QueueConnectionFactory");
/* 2 */
      Queue queue = (Queue)initialContext.lookup("java:comp/env/jms/Queue");
/* 3 */
      QueueConnection queueConnection =
        queueConnectionFactory.createQueueConnection();                    /*
4 */
      QueueSession queueSession =
        queueConnection.createQueueSession(false, Session.AUTO_ACKNOWLEDGE);
/* 5 */
      QueueSender queueSender = queueSession.createSender(queue);          /*
6 */
      queueSender.send(Message);                                           /*
7 */
      queueConnection.close();                                             /*
8 */
    } catch( Exception e ) {
      ...
    }
    ...
  }
}
```

1. Construct a JNDI start context.
2. Acquire a QueueConnectionFactory object (if the JNDI name is 'QueueConnectionFactory').
3. Acquire a Queue object (if the JNDI name is 'Queue').
4. Create QueueConnection.
5. Create QueueSession.
6. Create QueueSender.
7. Send a message.
8. Close QueueConnection.

Note

For information on the specification of an environment property when a JNDI start context is constructed, refer to 'Setting Details of JNDI Environment Properties' in Chapter 6.

Creating a Receiver

A receiver requests the event channel for a message. The following provides a procedure example and a processing flow in which a receiver requests the event channel for a message.

Subscriber

```
public class SubscriberS {
  public static void main() {
    ...
    try {
      InitialContext initialContext = new InitialContext();
* 1 */
      TopicConnectionFactory topicConnectionFactory = (TopicConnectionFactory)
        initialContext.lookup("java:comp/env/jms/TopicConnectionFactory");
    /* 2 */
      Topic topic = (Topic)initialContext.lookup("java:comp/env/jms/Topic");
/* 3 */
      TopicConnection topicConnection =
        topicConnectionFactory.createTopicConnection();
* 4 */
      TopicSession topicSession =
        topicConnection.createTopicSession(false, Session.AUTO_ACKNOWLEDGE);
   /* 5 */
      TopicSubscriber topicSubscriber = topicSession.createSubscriber(topic);
  /* 6 */
      topicConnection.start();                                          /*
7 */
      Message message = topicSubscriber.receive();                      /*
8 */
      topicConnection.close();                                          /*
9 */
    } catch( Exception e ) {
      ...
    }
    ...
  }
}
```

1. Construct a JNDI start context.
2. Acquire a QueueConnectionFactory object (if the JNDI name is 'QueueConnectionFactory').
3. Acquire a Queue object (if the JNDI name is 'Queue').
4. Create QueueConnection.
5. Create QueueSession.
6. Create QueueReceiver.
7. Start the delivery of a message upon connection.
8. Receive a message.
9. Close QueueConnection.

Note

For information on the specification of an environment property when a JNDI start context is constructed, refer to Setting Details of JNDI Environment Properties in Chapter 6.

Message Listener

This section describes developing a receiving application using Message Listener.

Message Listener is a function of automatically handling a message when it arrives at a receiver.

Creating a Subscriber using Message Listener

To handle a received message, register a Message Listener object. When a message arrives, Message Listener is invoked. The following provides a procedure example and a processing flow for receiving a message using Message Listener.

Subscriber using Message Listener

```
public class SubscriberA implements MessageListener {                          /*
1 */
  public static void main() {
    ...
    SubscriberA subscriber = new SubscriberA();                                 /*
2 */
    ...
    try {
      InitialContext initialContext = new InitialContext();                     /*
3 */
      TopicConnectionFactory topicConnectionFactory = (TopicConnectionFactory)
        initialContext.lookup("java:comp/env/jms/TopicConnectionFactory");
/* 4 */
      Topic topic = (Topic)initialContext.lookup("java:comp/env/jms/Topic");
/* 5 */
      TopicConnection topicConnection =
        topicConnectionFactory.createTopicConnection();                         /*
6 */
      TopicSession topicSession =
        topicConnection.createTopicSession(false, Session.AUTO_ACKNOWLEDGE);
/* 7 */
      TopicSubscriber topicSubscriber = topicSession.createSubscriber(topic);
/* 8 */
      topicSubscriber.setMessageListener(subscriber);                           /*
9 */
      topicConnection.start();                                                  /*
10 */
      /* Queuing processing */
      topicConnection.close();                                                  /*
11 */
    } catch( Exception e ) {
      ...
    }
    ...
  }
```

```
  public void onMessage(Message message) {
    ...
    try {
    } catch( JMSException e ) {
      ...
    }
    ...
  }
}
```

1. Implement a MessageListener interface.
2. Create a class instance.
3. Construct a JNDI start context.
4. Acquire a TopicConnectionFactory object (if the JNDI name is 'TopicConnectionFactory').
5. Acquire a Topic object (if the JNDI name is 'Topic').
6. Create TopicConnection.
7. Create TopicSession.
8. Create TopicSubscriber.
9. Register the MessageListener object.
10. Start the delivery of a message upon connection.
11. Close TopicConnection.

Note

For information on the specification of an environment property when a JNDI start context is constructed, refer to 'Setting Details of JNDI Environment Properties' in Chapter 6.

Creating a Receiver using Message Listener

To handle a received message, register a Message Listener object. When a message arrives, Message Listener is invoked. The following provides a procedure example and a processing flow for receiving a message using Message Listener.

Receiver using Message Listener

```
public class ReceiverA implements MessageListener {                    /*
1 */
  public static void main() {
    ...
    ReceiverA receiver = new ReceiverA();                               /*
2 */
    ...
    try {
      InitialContext initialContext = new InitialContext();             /*
3 */
      QueueConnectionFactory queueConnectionFactory = (QueueConnectionFactory)
        initialContext.lookup("java:comp/env/jms/QueueConnectionFactory");
```

```
/* 4 */
      Queue queue = (Queue)initialContext.lookup("java:comp/env/jms/Queue");
/* 5 */
      QueueConnection queueConnection =
        queueConnectionFactory.createQueueConnection();                    /*
6 */
      QueueSession queueSession =
        queueConnection.createQueueSession(false, Session.AUTO_ACKNOWLEDGE);
/* 7 */
      QueueReceiver queueReceiver = queueSession.createReceiver(queue);     /*
8 */
      queueReceiver.setMessageListener(receiver);                           /*
9 */
      queueConnection.start();                                              /*
10 */
      /* Queuing processing */
      queueConnection.close();                                              /*
11 */
    } catch( Exception e ) {
      ...
    }
    ...
  }


  public void onMessage(Message message) {
    ...
    try {
    } catch( JMSException e ) {
      ...
    }
    ...
  }
}
```

1. Implement a MessageListener interface.
2. Create a class instance.
3. Construct a JNDI start context.
4. Acquire a QueueConnectionFactory object (if the JNDI name is 'QueueConnectionFactory').
5. Acquire a Queue object (if the JNDI name is 'Queue').
6. Create QueueConnection.
7. Create QueueSession.
8. Create QueueReceiver.
9. Register the MessageListener object.
10. Start the delivery of a message upon connection.
11. Close QueueConnection.

Note

For information on the specification of an environment property when a JNDI start context is constructed, refer to 'Setting Details of JNDI Environment Properties' in Chapter 6.

Durable Subscription Function

This section describes developing a receiving application using the Durable Subscription function.

The Durable Subscription function is a function that allows a message sent while an application is not active to be received after the application becomes active.

Creating a Subscriber using the Durable Subscription Function

Create a durable subscriber. A durable subscriber requests the event channel for a message. The following provides a procedure example and a processing flow for receiving a message using the Durable Subscription function.

Subscriber using the Durable Subscription Function

```
public class SubscriberD {
  public static void main() {
    ...
    try {
      InitialContext initialContext = new InitialContext();                /*
1 */
      TopicConnectionFactory topicConnectionFactory = (TopicConnectionFactory)
        initialContext.lookup("java:comp/env/jms/TopicConnectionFactory");
/* 2 */
      Topic topic = (Topic)initialContext.lookup("java:comp/env/jms/Topic");
/* 3 */
      TopicConnection topicConnection =
        topicConnectionFactory.createTopicConnection();                    /*
4 */
      TopicSession topicSession =
        topicConnection.createTopicSession(false, Session.AUTO_ACKNOWLEDGE);
/* 5 */
      TopicSubscriber topicSubscriber =
        topicSession.createDurableSubscriber(topic, "dsub");               /*
6 */
      topicConnection.start();                                             /*
7 */
      Message message = topicSubscriber.receive();                         /*
8 */
      topicSubscriber.close();                                             /*
9 */
      topicSession.unsubscribe("dsub");                                    /*
10 */
      topicConnection.close();                                             /*
```

```
11 */
    } catch( Exception e ) {
      ...
    }
    ...
  }
}
```

1. Construct a JNDI start context.
2. Acquire a TopicConnectionFactory object (if the JNDI name is 'TopicConnectionFactory').
3. Acquire a Topic object (if the JNDI name is 'Topic').
4. Create TopicConnection.
5. Create TopicSession.
6. Create Durable TopicSubscriber (if the Durable Subscription name is 'dsub').
7. Start the delivery of a message upon connection.
8. Receive a message.
9. Close Durable TopicSubscriber.
10. Release Durable TopicSubscriber (if messages that were sent during interruption of the application are not received after restarting).
11. Close TopicConnection.

Notes

For information on the specification of an environment property when a JNDI start context is constructed, refer to 'Setting Details of JNDI Environment Properties' in Chapter 6.

When Durable TopicSubscriber is released, the connection state maintained by the JMS provider in place of the application is deleted, so messages that were sent during interruption of the application cannot be received after restarting of the application. Release Durable TopicSubscriber only when the Durable Subscription function is no longer required.

Note on using the Durable Subscription Function

A durable subscriber is identified using the Durable Subscription name and a client identifier. To start up multiple applications on the same machine, either use a different Durable Subscription name for each of them or use a ConnectionFactory definition that specifies a different client identifier.

Message Priority and Lifetime

The priority and lifetime of a message can be specified only by the sender of the message.

Such a specification can be made using the publish(Message message, int deliveryMode, int priority, long timeToLive) method of the TopicPublisher interface or the send (Message message, int deliveryMode, int priority, long timeToLive) method for the QueueSender interface.

If this method is not used, the default priority and lifetime is assumed.

By default, the priority is 4 and the lifetime is endless. These settings can be changed using setPriority(int defaultPriority) and setTimeToLive(long timeToLive) of the MessageProducer interface.

- As the priority, specify in priority a value from 0 to 9 in the order of increasing priority.
- As the lifetime, specify in timeToLive time in milliseconds.

Note

Although time in milliseconds is supported in timeToLive, the message timeout time of an event channel is specified only in seconds. Thus, the timeToLive value is rounded to the nearest value.

Message Persistent Function

To use the message persistent function, you need to:

For Publish/Subscribe Messaging Model

1. Use the *esmkchnl* command to create a persistent channel.
2. Specify javax.jms.DeliveryMode.PERSISTENT in argument deliveryMode of the publish method of the TopicPublisher interface on the publisher side.

For Point-To-Point Messaging Model

1. Use the *esmkchnl* command of specifying -ptp option to create a persistent channel.
2. Specify javax.jms.DeliveryMode.PERSISTENT in argument deliveryMode of the send method of the QueueSender interface on the sender side.

Local Transaction

Creating a Publisher using a Local Transaction

Publisher in a Local Transaction

```
public class Publisher {
  public static void main() {
    ...
    try {
      InitialContext initialContext = new InitialContext();                /*
1 */
      TopicConnectionFactory topicConnectionFactory = (TopicConnectionFactory)
        initialContext.lookup("java:comp/env/jms/TopicConnectionFactory");
```

```
/* 2 */
      Topic topic = (Topic)initialContext.lookup("java:comp/env/jms/Topic");
/* 3 */
      TopicConnection topicConnection =
        topicConnectionFactory.createTopicConnection();                    /*
4 */
      TopicSession topicSession = topicConnection.createTopicSession(true, 0);
/* 5 */
      TopicPublisher topicPublisher = topicSession.createPublisher(topic);
/* 6 */
      topicPublisher.publish(Message);                                     /*
7 */
      topicSession.commit();                                               /*
8 */
      topicConnection.close();                                             /*
9 */
    } catch( Exception e ) {
      ...
    }
    ...
  }
}
```

1. Construct a JNDI start context.
2. Acquire a TopicConnectionFactory object (if the JNDI name is 'TopicConnectionFactory').
3. Acquire a Topic object (if the JNDI name is 'Topic').
4. Create TopicConnection.
5. Create TopicSession.
6. Create TopicPublisher.
7. Send a message.
8. Make commitment.
9. Close TopicConnection.

Note

For information on the specification of an environment property when a JNDI start context is constructed, refer to 'Setting Details of JNDI Environment Properties' in Chapter 6.

Creating a Subscriber using a Local Transaction

Subscriber in a Local Transaction

```
public class SubscriberS {
  public static void main() {
    ...
    try {
      InitialContext initialContext = new InitialContext();                /*
1 */
```

```
      TopicConnectionFactory topicConnectionFactory = (TopicConnectionFactory)
        initialContext.lookup("java:comp/env/jms/TopicConnectionFactory");
/* 2 */
      Topic topic = (Topic)initialContext.lookup("java:comp/env/jms/Topic");
/* 3 */
      TopicConnection topicConnection =
        topicConnectionFactory.createTopicConnection();                    /*
4 */
      TopicSession topicSession = topicConnection.createTopicSession(true, 0);
/* 5 */
      TopicSubscriber topicSubscriber = topicSession.createSubscriber(topic);
/* 6 */
      topicConnection.start();                                            /*7
*/
      Message message = topicSubscriber.receive();                         /*
8 */
      topicSession.commit();                                               /*
9 */
      topicConnection.close();                                             /*
10 */
    } catch( Exception e ) {
      ...
    }
    ...
  }
}
```

1. Construct a JNDI start context.
2. Acquire a TopicConnectionFactory object (if the JNDI name is 'TopicConnectionFactory').
3. Acquire a Topic object (if the JNDI name is 'Topic').
4. Create TopicConnection.
5. Create TopicSession.
6. Create TopicSubscriber.
7. Start the delivery of a message upon connection.
8. Receive a message.
9. Make commitment.
10. Close TopicConnection.

Note

For information on the specification of an environment property when a JNDI start context is constructed, refer to 'Setting Details of JNDI Environment Properties' in Chapter 6.

Creating a Sender using a Local Transaction

Sender in a Local Transaction

```
public class Sender {
  public static void main() {
    ...
    try {
      InitialContext initialContext = new InitialContext();
/* 1 */
      QueueConnectionFactory queueConnectionFactory = (QueueConnectionFactory)
        initialContext.lookup("java:comp/env/jms/QueueConnectionFactory");
     /* 2 */
      Queue queue = (Queue)initialContext.lookup("java:comp/env/jms/Queue");
   /* 3 */
      QueueConnection queueConnection =
        queueConnectionFactory.createQueueConnection();
/* 4 */
      QueueSession queueSession = queueConnection.createQueueSession(true, 0);
 /* 5 */
      QueueSender queueSender = queueSession.createSender(queue);
/* 6 */
      queueSender.send(Message);
/* 7 */
      queueSession.commit();
/* 8 */
      queueConnection.close();
/* 9 */
    } catch( Exception e ) {
      ...
    }
    ...
  }
}
```

1. Construct a JNDI start context.
2. Acquire a QueueConnectionFactory object (if the JNDI name is 'QueueConnectionFactory').
3. Acquire a Queue object (if the JNDI name is 'Queue').
4. Create QueueConnection.
5. Create QueueSession.
6. Create QueueSender.
7. Send a message.
8. Make commitment.
9. Close QueueConnection.

Note

For information on the specification of an environment property when a JNDI start context is constructed, refer to 'Setting Details of JNDI Environment Properties' in Chapter 6.

Creating a Receiver using a Local Transaction

Receiver in a Local Transaction

```
public class ReceiverS {
  public static void main() {
    ...
    try {
      InitialContext initialContext = new InitialContext();                /*
1 */
      QueueConnectionFactory queueConnectionFactory = (QueueConnectionFactory)
        initialContext.lookup("java:comp/env/jms/QueueConnectionFactory");
/* 2 */
      Queue queue = (Queue)initialContext.lookup("java:comp/env/jms/Queue");
/* 3 */
      QueueConnection queueConnection =
        queueConnectionFactory.createQueueConnection();                    /*
4 */
      QueueSession queueSession = queueConnection.createQueueSession(true, 0);
/* 5 */
      QueueReceiver queueReceiver = queueSession.createReceiver(queue);     /*
6 */
      queueConnection.start();                                             /*
7 */
      Message message = queueReceiver.receive();                           /*
8 */
      queueSession.commit();                                               /*
9 */
      queueConnection.close();                                             /*
10 */
    } catch( Exception e ) {
      ...
    }
    ...
  }
}
```

1. Construct a JNDI start context.
2. Acquire a QueueConnectionFactory object (if the JNDI name is 'QueueConnectionFactory').
3. Acquire a Queue object (if the JNDI name is 'Queue').
4. Create QueueConnection.
5. Create QueueSession.
6. Create QueueReceiver.
7. Start the delivery of a message upon connection.

8. Receive a message.
9. Make commitment.
10. Close QueueConnection.

Note

For information on the specification of an environment property when a JNDI start context is constructed, refer to 'Setting Details of JNDI Environment Properties' in Chapter 6.

Global Transaction

Creating a Publisher using a Global Transaction

Publisher in a Global Transaction

```
public class Publisher {
  public static void main() {
    ...
    try {
      InitialContext initialContext = new InitialContext();                /*
1 */
      TopicConnectionFactory topicConnectionFactory = (TopicConnectionFactory)
        initialContext.lookup("java:comp/env/jms/TopicConnectionFactory");
/* 2 */
      Topic topic = (Topic)initialContext.lookup("java:comp/env/jms/Topic");
/* 3 */
      javax.transaction.UserTransaction ut =
(javax.transaction.UserTransaction)
        initialContext.lookup("java:comp/UserTransaction");                /*
4 */
      TopicConnection topicConnection =
        topicConnectionFactory.createTopicConnection();                    /*
5 */
      TopicSession topicSession = topicConnection.createTopicSession(true ,
 0);/* 6 */
      TopicPublisher topicPublisher = topicSession.createPublisher(topic);
/* 7 */
      ut.begin();                                                          /*
8 */
      topicPublisher.publish(Message);                                     /*
9 */
      ut.commit();                                                         /*
10 */
      topicConnection.close();                                             /*
11 */
    } catch( Exception e ) {
      ...
    }
    ...
  }
}
```

1. Construct a JNDI start context.
2. Acquire a TopicConnectionFactory object (if the JNDI name is 'TopicConnectionFactory').
3. Acquire a Topic object (if the JNDI name is 'Topic').
4. Acquire a user transaction object.
5. Create TopicConnection.
6. Create TopicSession.
7. Create TopicPublisher.
8. Start a global transaction.
9. Send a message.
10. Complete the global transaction.
11. lose TopicConnection.

Note

For information on the specification of an environment property when a JNDI start context is constructed, refer to 'Setting Details of JNDI Environment Properties' in Chapter 6.

Creating a Subscriber using a Global Transaction

Subscriber in a Global Transaction

```
public class SubscriberS {
  public static void main() {
    ...
    try {
      InitialContext initialContext = new InitialContext();                /*
1 */
      TopicConnectionFactory topicConnectionFactory = (TopicConnectionFactory)
        initialContext.lookup("java:comp/env/jms/TopicConnectionFactory");
/* 2 */
      Topic topic = (Topic)initialContext.lookup("java:comp/env/jms/Topic");
/* 3 */
      javax.transaction.UserTransaction ut =
(javax.transaction.UserTransaction)
        initialContext.lookup("java:comp/UserTransaction");                /*
4 */
      TopicConnection topicConnection =
        topicConnectionFactory.createTopicConnection();                    /*
5 */
      TopicSession topicSession = topicConnection.createTopicSession(true ,
0);/* 6 */
      TopicSubscriber topicSubscriber = topicSession.createSubscriber(topic);
/* 7 */
      topicConnection.start();                                             /*
8 */
      ut.begin();                                                          /*
9 */
      Message message = topicSubscriber.receive();                         /*
10 */
```

```
      ut.commit();                                                    /*
11 */
      topicConnection.close();                                        /*
12 */
    } catch( Exception e ) {
      ...
    }
    ...
  }
}
```

1. Construct a JNDI start context.
2. Acquire a TopicConnectionFactory object (if the JNDI name is 'TopicConnectionFactory').
3. Acquire a Topic object (if the JNDI name is 'Topic').
4. Acquire a user transaction object.
5. Create TopicConnection.
6. Create TopicSession.
7. Create TopicSubscriber.
8. Start the delivery of a message upon connection.
9. Start a global transaction.
10. Receive a message.
11. Complete the global transaction.
12. Close TopicConnection.

Notes

For information on the specification of an environment property when a JNDI start context is constructed, refer to 'Setting Details of JNDI Environment Properties' in Chapter 6.

When the Durable Subscription function is used, releasing (unsubscribe) the Durable Subscription can be specified between start (begin) of global transaction and end (commit) of the global transaction.

Creating a Sender using a Global Transaction

Sender in a Global Transaction

```
public class Sender {
  public static void main() {
    ...
    try {
      InitialContext initialContext = new InitialContext();           /*
1 */
      QueueConnectionFactory queueConnectionFactory = (QueueConnectionFactory)
        initialContext.lookup("java:comp/env/jms/QueueConnectionFactory");
/* 2 */
      Queue queue = (Queue)initialContext.lookup("java:comp/env/jms/Queue");
```

```
/* 3 */
      javax.transaction.UserTransaction ut =
(javax.transaction.UserTransaction)
        initialContext.lookup("java:comp/UserTransaction");               /*
4 */
      QueueConnection queueConnection =
        queueConnectionFactory.createQueueConnection();                   /*
5 */
      QueueSession queueSession = queueConnection.createQueueSession(true ,
0);/* 6 */
      QueueSender queueSender = queueSession.createSender(queue);         /*
7 */
      ut.begin();                                                         /*
8 */
      queueSender.send(Message);                                          /*
9 */
      ut.commit();                                                        /*
10 */
      queueConnection.close();                                            /*
11 */
    } catch( Exception e ) {
      ...
    }
    ...
  }
}
```

1. Construct a JNDI start context.
2. Acquire a QueueConnectionFactory object (if the JNDI name is 'QueueConnectionFactory').
3. Acquire a Queue object (if the JNDI name is 'Queue').
4. Acquire a user transaction object.
5. Create QueueConnection.
6. Create QueueSession.
7. Create QueueSender.
8. Start a global transaction.
9. Send a message.
10. Complete the global transaction.
11. Close QueueConnection.

Note

For information on the specification of an environment property when a JNDI start context is constructed, refer to 'Setting Details of JNDI Environment Properties' in Chapter 6.

Creating a Receiver using a Global Transaction

Receiver in a Global Transaction

```
public class ReceiverS {
  public static void main() {
    ...
    try {
      InitialContext initialContext = new InitialContext();                /*
1 */
      QueueConnectionFactory queueConnectionFactory = (QueueConnectionFactory)
        initialContext.lookup("java:comp/env/jms/QueueConnectionFactory");
/* 2 */
      Queue queue = (Queue)initialContext.lookup("java:comp/env/jms/Queue");
/* 3 */
      javax.transaction.UserTransaction ut =
(javax.transaction.UserTransaction)
        initialContext.lookup("java:comp/UserTransaction");                /*
4 */
      QueueConnection queueConnection =
        queueConnectionFactory.createQueueConnection();                    /*
5 */
      QueueSession queueSession = queueConnection.createQueueSession(true ,
0);/* 6 */
      QueueReceiver queueReceiver = queueSession.createReceiver(queue);     /*
7 */
      queueConnection.start();                                             /*
8 */
      ut.begin();                                                          /*
9 */
      Message message = queueReceiver.receive();                           /*
10 */
      ut.commit();                                                         /*
11 */
      queueConnection.close();                                             /*
12 */
    } catch( Exception e ) {
      ...
    }
    ...
  }
}
```

1. Construct a JNDI start context.
2. Acquire a QueueConnectionFactory object (if the JNDI name is 'QueueConnectionFactory').
3. Acquire a Queue object (if the JNDI name is 'Queue').
4. Acquire a user transaction object.
5. Create QueueConnection.
6. Create QueueSession.
7. Create QueueReceiver.

8. Start the delivery of a message upon connection.
9. Start a global transaction.
10. Receive a message.
11. Complete the global transaction.
12. Close QueueConnection.

Note

For information on the specification of an environment property when a JNDI start context is constructed, refer to 'Setting Details of JNDI Environment Properties' in Chapter 6.

Note on Starting an Application

Before starting an application, specify True in the following environment property:

com.fujitsu.ObjectDirector.CORBA.GlobalTransactionMode

For information on the specification of an environment property, refer to 'Setting Details of JNDI Environment Properties' in Chapter 6.

The following shows a specification example (specification in a command line).

```
java -Dcom.fujitsu.ObjectDirector.CORBA.GlobalTransactionMode=True Publisher
```

Linkage with a CORBA Application

Communication from a JMS Application to a CORBA Application

During communication from a JMS application to a CORBA application via the notification service, a JMS Message is converted to StructuredEvent.

Interstage JMS supports BytesMessage and TextMessage as the JMS message types that can be linked with a CORBA application.

The following shows the conversion of a JMS Message to StructuredEvent.

Table 17-1 Conversion of JMS Message to Structured Event

Location	JMS Message	StructuredEvent
Header	JMSPriority JMSExpiration	QoS property Priority Timeout FixedHeader: Automatically added by JMS-Provider domain_name='JMS' type_name='v5.0' event_name=''
Property	Not converted	

Location	JMS Message	StructuredEvent
Body	BytesMessage	Should be retrieved as a sequence-type of octet-type.
	TextMessage	Should be retrieved as a wstring-type.

Correspondence between JMSPriority and QoS property priority.

JMSPriority:	0	1	2	3	4	5	6	7	8	9
Priority(QoS):	-3	-3	-2	-1	0	1	2	2	3	3

Communication from a CORBA Application to a JMS Application

During communication from a CORBA application to a JMS application via the notification service, only conversion from remainder_of_body of StructuredEvent to the body of a JMS Message is enabled.

Interstage JMS supports the following StructuredEvent data types that can be linked with a CORBA application.

Table 17-2 StructuredEvent Data Types

StructuredEvent	JMS Message
string-type	TextMessage
wstring-type	
array of octet-type	BytesMessage
sequence-type of octet-type	

Message Selector Function

To use the message selector function, you need to create the following applications:

(1) Sending Application

The send application specifies a property value that corresponds to the message selector when a message is created.

The following examples show how to set the property value 'FUJITSU' and property name 'NAME' in a message.

```
String name = "FUJITSU";
message.setStringProperty("NAME", name);
```

(2) Receiving Application

The receive application specifies a message selector statement as a parameter when a Subscriber, Durable Subscriber, Receiver, or Browser is created. A message selector statement is a query character string to be used in the WHERE clause of an SQL statement.

The following example shows how to set up a message selector to receive messages with the property name 'NAME' and the property value 'FUJITSU'.

```
String selector = "NAME = 'FUJITSU'";
topicsession.createSubscriber(topic, selector);
```

Message Selector Conditional Expression

Conditional expressions follow the format to be used in the WHERE clause of an SQL statement in SQL-92.

```
color = 'blue'
```

The element 'color' on the left side in the above conditional expression is called an identifier. The identifier is compared with the property name of JMS Messages during filtering.

The element 'blue' on the right side is called a literal. The literal is compared with the property value of JMS Messages during filtering.

Literals come in three types: Character string literal, exact numeric literal, and approximate numeric literal.

A character string literal must be enclosed in single quotation marks.

An exact numeric literal is a value without a decimal place such as 57, -957, and +62.

An approximate numeric literal is either a value with an exponential part such as 7E3 and -57.9E2 or a value with a decimal place such as 7., -95.7, and +6.2.

For more information on the syntax of a conditional expression, refer to 'Interface Message in Package javax.jms', in the Java documentation.

The following shows examples of conditional expressions.

- **Comparison operation conditional expression**

  The following comparison operators can be used.

  = (is equal to)

  > (is greater than)

  >= (is equal to or greater than)

  < (is less than)

  <= (is equal to or less than)

  <> (is not equal to)

Example

JMS Messages where the property with property name NAME has a property value 'FUJITSU' can be received.

```
NAME = 'FUJITSU'
```

Example

JMS Messages where the property with property name NUMBER has a property value of 1000 or more can be received.

```
NUMBER >= 1000
```

Example

JMS Messages where the property with property name NUMBER has a property value of 230 or less can be received.

```
NUMBER <= 10 * 20 + 30
```

As shown in these examples, arithmetic operators (+, -, *, and /) can be used.

- **BETWEEN conditional expression**

  A BETWEEN conditional expression allows you to perform searches over a range.

  Example

  JMS Messages where the property with property name NUMBER has a property value between (and including) 100 and 1000 can be received.

  ```
  NUMBER BETWEEN 100 AND 1000
  ```

  Example

  JMS Messages where the property with property name NUMBER has a property value less than 100 or more than 1000 can be received.

  ```
  NUMBER NOT BETWEEN 100 AND 1000
  ```

- **LIKE conditional expression**

  A LIKE conditional expression allows to perform searches using pattern search.

  Specify '%' and '_' in a character string literal to perform pattern search.

  '_' represents any character and '%' represents any character string.

  An escape character that is an option is an independent character literal used to handle '_' or '%' merely as a character string.

Example

JMS Messages where the property with property name PROPERTY has a property value of the form 'any character(s) + C' can be received.

```
PROPERTY LIKE '%C'
```

JMS Messages can be received if the property value is ABC, CCC, or C but cannot be received if the property value is AB.

Example

JMS Messages where the property with property name PROPERTY has a property value that is not of the form 'any character(s) + C' can be received.

```
PROPERTY NOT LIKE '%C'
```

JMS Messages can be received if the property value is AB but cannot be received if the property value is ABC, CCC, or C.

Example

JMS Messages where the property with property name PROPERTY has a property value 'any one character + C' can be received.

```
PROPERTY LIKE '_C'
```

JMS Messages can be received if the property value is AC or CC but cannot be received if the property value is ABC or C.

Example

JMS Messages where the property with property name PROPERTY has a property value of the form 'any character(s) + % + C' can be received.

```
PROPERTY LIKE '%#%C' ESCAPE '#'
```

JMS Messages can be received if the property value is A%C but cannot be received if the property value is AAC.

- **NULL conditional expression**

  The NULL conditional expression allows to perform searches depending on whether a property is present.

  Example

  JMS Messages without property name PROPERTY can be received.

  ```
  PROPERTY IS NULL
  ```

Example

JMS Messages with property name PROPERTY can be received.

```
PROPERTY IS NOT NULL
```

- **IN conditional expression**

  The IN conditional expression allows to search items in a list.

  Example

  JMS Messages with where the property with property name PROPERTY has the property value 'AAA', 'BBB' or 'CCC' can be received.

```
PROPERTY IN ('AAA','BBB','CCC')
```

  Example

  JMS Messages where the property with property name PROPERTY has a property value other than 'AAA', 'BBB' or 'CCC' can be received.

```
PROPERTY NOT IN ('AAA','BBB','CCC')
```

- **Mixture of conditional expressions**

  A mixture of the above conditional expressions can be specified using NOT, AND, or OR.

  Example

  JMS Messages where the property with property name NUMBER has a property value between (and including) 100 and 1000 OR the property with property name PROPERTY has a property value of the form 'any character(s) + C' can be received.

```
(NUMBER BETWEEN 100 AND 1000) OR (PROPERTY LIKE '%C')
```

  Example

  JMS Messages where the property with property name NUMBER has a property value between (and including) 100 and 1000 AND the property with property name PROPERTY has a property value of the form 'any character(s) + C' can be received.

```
(NUMBER BETWEEN 100 AND 1000) AND (PROPERTY LIKE '%C')
```

Example

JMS Messages where the property with property name NUMBER has a property value between (and including) 100 and 1000 OR the property with property name PROPERTY has a property value other than of the form 'any character(s) + C' can be received.

```
NOT ((NUMBER BETWEEN 100 AND 1000) AND (PROPERTY LIKE '%C'))
```

Notes

- A message selector statement with a length up to 4096 bytes can be specified.
- An identifier or character string literal with a length up to 1024 bytes can be specified in a conditional expression.
- Up to a total of 512 identifiers and character string literals can be specified in a conditional expression.
- Up to 256 lists can be specified in an IN conditional expression.

Queue Browser Function

In the application, create a browser to reference messages accumulated in a queue.

After that, the messages can be retrieved in sequence and the contents of the queue can be browsed.

The following shows an example of the required procedures and the processing flow for a queue browser.

Browser

```
public class Browser {
  public static void main() {
    ...
    try {
      InitialContext initialContext = new InitialContext();                /*
1 */
      QueueConnectionFactory queueConnectionFactory = (QueueConnectionFactory)
        initialContext.lookup("java:comp/env/jms/QueueConnectionFactory");
/* 2 */
      Queue queue = (Queue)initialContext.lookup("java:comp/env/jms/Queue");
/* 3 */
      QueueConnection queueConnection =
        queueConnectionFactory.createQueueConnection();                    /*
4 */
      QueueSession queueSession =
        queueConnection.createQueueSession(false, Session.AUTO_ACKNOWLEDGE);
/* 5 */
      QueueBrowser queueBrowser = queueSession.createBrowser(queue);       /*
6 */
      queueConnection.start();                                             /*
7 */
      java.util.Enumeration e = qBrowser.getEnumeration();                 /*
8 */
      Message message;
```

```
      while ( e.hasMoreElements() ) {
        message = e.nextElement();                                    /*
9 */
        ...
      }
      queueBrowser.close();                                           /*
10 */
      queueConnection.close();                                        /*
11 */
    } catch( Exception e ) {
      ...
    }
    ...
  }
}
```

1. Construct a JNDI start context.
2. Acquire a QueueConnectionFactory object (if the JNDI name is 'QueueConnectionFactory').
3. Acquire a Queue object (if the JNDI name is 'Queue').
4. Create QueueConnection.
5. Create QueueSession.
6. Create QueueBrowser.
7. Start the delivery of a message upon connection.
8. Start browsing.
9. Acquire one message.
10. Close QueueBrowser.
11. Close QueueConnection.

Notes

For information on the specification of an environment property when a JNDI start context is constructed, refer to 'Setting Details of JNDI Environment Properties' in Chapter 6.

Only one queue browser can be used per channel.

Notes on Using TopicRequestor/QueueRequestor

The request method of the TopicRequestor class or the QueueRequestor class sends a request message to Topic or Queue and waits for the response. Depending on the application configuration, the request method does not respond when the receiver application is not connected or the receiver application does not return the response message.

To avoid the indefinite wait in the queue described above, the system property can be used for instruct the response from the request method if no response is returned from the method in a certain time.

System property name

com.fujitsu.interstage.jms.receive_timeout

Specification

Set the timeout time in units of milliseconds.

Example

```
-Dcom.fujitsu.interstage.jms.receive_timeout=10000
```

Note

If a value smaller than the -wtime set value of the Event Service operating environment is specified, the queuing time is set to -wtime.

If a timeout occurs, the request method returns null.

Interface

This section describes JMS APIs supported by Interstage JMS.

API lists of the package javax.jms are shown below.

For information on APIs, refer to Package javax.jms in the Java docs.

API List of the Package javax.jms (Part 1)

Table 17-3 API List of javax.jms

Interface name/Class name	Method	Support
BytesMessage	readBoolean()	O
	readByte()	O
	readBytes(byte[] value)	O
	readBytes(byte[] value, int length)	O
	readChar()	O
	readDouble()	O
	readFloat()	O
	readInt()	O
	readLong()	O
	readShort()	O
	readUnsignedByte()	O
	readUnsignedShort()	O
	readUTF()	O
	reset()	O
	writeBoolean(boolean value)	O
	writeByte(byte value)	O
	writeBytes(byte[] value)	O

Interface name/Class name	Method	Support
	writeBytes(byte[] value, int offset, int length)	O
	writeChar(char value)	O
	writeDouble(double value)	O
	writeFloat(float value)	O
	writeInt(int value)	O
	writeLong(long value)	O
	writeObject(java.lang.Object value)	O
	writeShort(short value)	O
	writeUTF(java.lang.String value)	O
Connection	close()	O
	getClientID()	O
	getExceptionListener()	O
	getMetaData()	O
	setClientID(java.lang.String clientID)	O
	setExceptionListener(ExceptionListener listener)	O
	start()	O
	stop()	O
ConnectionConsumer	close()	O
	getServerSessionPool()	O
ConnectionFactory	No method	

O: Supported

X: Not supported

API List of the Package javax.jms (Part 2)

Table 17-4 API List of javax.jms

Interface name/Class name	Method	Support
ConnectionMetaData	getJMSMajorVersion()	O
	getJMSMinorVersion()	O
	getJMSProviderName()	O (See Note)
	getJMSVersion()	O
	getJMSXPropertyNames()	O
	getProviderMajorVersion()	O
	getProviderMinorVersion()	O
	getProviderVersion()	O
DeliveryMode	No method	
Destination	No method	
ExceptionListener	onException(JMSException exception)	O

Interface name/Class name	Method	Support
MapMessage	getBoolean(java.lang.String name)	O
	getByte(java.lang.String name)	O
	getBytes(java.lang.String name)	O
	getChar(java.lang.String name)	O
	getDouble(java.lang.String name)	O
	getFloat(java.lang.String name)	O
	getInt(java.lang.String name)	O
	getLong(java.lang.String name)	O
	getMapNames()	O
	getObject(java.lang.String name)	O
	getShort(java.lang.String name)	O
	getString(java.lang.String name)	O
	itemExists(java.lang.String name)	O
	setBoolean(java.lang.String name, boolean value)	O
	setByte(java.lang.String name, byte value)	O
	setBytes(java.lang.String name, byte[] value)	O
	setBytes(java.lang.String name, byte[] value, int offset, int length)	O
	setChar(java.lang.String name, char value)	O
	setDouble(java.lang.String name, double value)	O
	setFloat(java.lang.String name, float value)	O
	setInt(java.lang.String name, int value)	O
	setLong(java.lang.String name, long value)	O
	setObject(java.lang.String name, java.lang.Object value)	O
	setShort(java.lang.String name, short value)	O
	setString(java.lang.String name, java.lang.String value)	O

O: Supported

X: Not supported

Note: 'Interstage Application Server' is returned.

API List of the Package javax.jms (Part 3)

Table 17-5 API List of javax.jms

Interface name/Class name	Method	Support
Message	acknowledge()	O
	clearBody()	O
	clearProperties()	O
	getBooleanProperty(java.lang.String name)	O
	getByteProperty(java.lang.String name)	O
	getDoubleProperty(java.lang.String name)	O
	getFloatProperty(java.lang.String name)	O
	getIntProperty(java.lang.String name)	O
	getJMSCorrelationID()	O
	getJMSCorrelationIDAsBytes()	X
	getJMSDeliveryMode()	O
	getJMSDestination()	O
	getJMSExpiration()	O
	getJMSMessageID()	O
	getJMSPriority()	O
	getJMSRedelivered()	O
	getJMSReplyTo()	O
	getJMSTimestamp()	O
	getJMSType()	O
	getLongProperty(java.lang.String name)	O
	getObjectProperty(java.lang.String name)	O
	getPropertyNames()	O
	getShortProperty(java.lang.String name)	O
	getStringProperty(java.lang.String name)	O
	propertyExists(java.lang.String name)	O

Interface name/Class name	Method	Support
	setBooleanProperty(java.lang.String name, boolean value)	O
	setByteProperty(java.lang.String name, byte value)	O
	setDoubleProperty(java.lang.String name, double value)	O
	setFloatProperty(java.lang.String name, float value)	O
	setIntProperty(java.lang.String name, int value)	O
	setJMSCorrelationID(java.lang.String correlationID)	O
	setJMSCorrelationIDAsBytes(byte[] correlationID)	X
	setJMSDeliveryMode(int deliveryMode)	O
	setJMSDestination(Destination destination)	O
	setJMSExpiration(long expiration)	O
	setJMSMessageID(java.lang.String id)	O
	setJMSPriority(int priority)	O
	setJMSRedelivered(boolean redelivered)	O
	setJMSReplyTo(Destination replyTo)	O
	setJMSTimestamp(long timestamp)	O
	setJMSType(java.lang.String type)	O
	setLongProperty(java.lang.String name, long value)	O
	setObjectProperty(java.lang.String name, java.lang.Object value)	O
	setShortProperty(java.lang.String name, short value)	O
	setStringProperty(java.lang.String name, java.lang.String value)	O

O: Supported

X: Not supported

API List of the Package javax.jms (Part 4)

Table 17-6 API List of javax.jms

Interface name/Class name	Method	Support
MessageConsumer	close()	O
	getMessageListener()	O
	getMessageSelector()	O
	receive()	O
	receive(long timeout)	O (Note 1)
	receiveNoWait()	O (Note 2)
	setMessageListener(MessageListener listener)	O
MessageListener	onMessage(Message message)	O
MessageProducer	close()	O
	getDeliveryMode()	O
	getDisableMessageID()	O
	getDisableMessageTimestamp()	O
	getPriority()	O
	getTimeToLive()	O
	setDeliveryMode(int deliveryMode)	O (Note 3)
	setDisableMessageID(boolean value)	O
	setDisableMessageTimestamp(boolean value)	O
	setPriority(int defaultPriority)	O
	setTimeToLive(long timeToLive)	O (Note 4)
ObjectMessage	getObject()	O
	setObject(java.io.Serializable object)	O

O: Supported

X: Not supported

Note 1) The timeout for receiving messages specified by receive() is checked at the intervals set for the wait time for the event channel event data. Therefore, the timeout for receiving messages is returned as follows:

- For 'Wait time for event data >= Value of receive()'

  The timeout for receiving messages is returned as 'Wait time for event data'.

  Example

  Wait time for event data: '40 seconds', Value of receive() '10': 40 seconds

- For 'Wait time for event data < Value of receive()'

  The timeout for receiving messages is returned as 'Wait time for event data * Number of checks'.

  Example

  Wait time for event data: '40 seconds', Value of receive() '50': 80 (40 * 2) seconds

Configure operations to not wait for event data using the receiveNoWait() method instead of the receive() method if performing operations that do not wait for messages.

Note 2) This method immediately returns null when there is no message which can be received.

Note 3) Only the same mode as the delivery mode of the event channel is supported.

Note 4) timeToLive supports in milliseconds. However, since the message timeout of the event channel is in seconds, timeToLive is rounded off to the nearest value.

API List of the Package javax.jms (Part 5)

Table 17-7 API List of javax.jms

Interface name/Class name	Method	Support
Queue	getQueueName()	O
	toString()	O (Note 1)
QueueBrowser	close()	O
	getEnumeration()	O
	getMessageSelector()	O
	getQueue()	O
QueueConnection	createConnectionConsumer(Queue queue, java.lang.String messageSelector, ServerSessionPool sessionPool, int maxMessages)	O
	createQueueSession(boolean transacted, int acknowledgeMode)	O (Note 2)
QueueConnectionFactory	createQueueConnection()	O
	createQueueConnection(java.lang.String userName, java.lang.String password)	O
QueueReceiver	getQueue()	O

Interface name/Class name	Method	Support
QueueRequestor	close()	O
	request(Message message)	O
QueueSender	getQueue()	O
	send(Message message)	O
	send(Message message, int deliveryMode, int priority, long timeToLive)	O (Note 3)
	send(Queue queue, Message message)	O
	send(Queue queue, Message message, int deliveryMode, int priority, long timeToLive)	O (Note 3)
QueueSession	createBrowser(Queue queue)	O
	createBrowser(Queue queue, java.lang.String messageSelector)	O
	createQueue(java.lang.String queueName)	O
	createReceiver(Queue queue)	O
	createReceiver(Queue queue, java.lang.String messageSelector)	O
	createSender(Queue queue)	O
	createTemporaryQueue()	O

O: Supported

X: Not supported

Note 1) 'com.fujitsu.interstage.jms:<Queue name>::<the group name of the event channel>::<the channel name of the event channel>' is returned.

Note 2) userName and password are ignored.

Note 3) timeToLive supports in milliseconds. However, since the message timeout of the event channel is in seconds, timeToLive is rounded off to the nearest value.

API List of the Package javax.jms (Part 6)

Table 17-8 API List of javax.jms

Interface name/Class name	Method	Support
ServerSession	getSession()	O
	start()	O
ServerSessionPool	getServerSession()	O
Session	close()	O
	commit()	O
	createBytesMessage()	O
	createMapMessage()	O
	createMessage()	O
	createObjectMessage()	O
	createObjectMessage(java.io.Serializable object)	O
	createStreamMessage()	O
	createTextMessage()	O
	createTextMessage(java.lang.String text)	O
	getMessageListener()	O
	getTransacted()	O
	recover()	O
	rollback()	O
	run()	O
	setMessageListener(MessageListener listener)	O (Note 1)

Interface name/Class name	Method	Support
StreamMessage	readBoolean()	O
	readByte()	O
	readBytes(byte[] value)	O
	readChar()	O
	readDouble()	O
	readFloat()	O
	readInt()	O
	readLong()	O
	readObject()	O
	readShort()	O
	readString()	O
	reset()	O
	writeBoolean(boolean value)	O
	writeByte(byte value)	O
	writeBytes(byte[] value)	O
	writeBytes(byte[] value, int offset, int length)	O
	writeChar(char value)	O
	writeDouble(double value)	O
	writeFloat(float value)	O
	writeInt(int value)	O
	writeLong(long value)	O
	writeObject(java.lang.Object value)	O
	writeShort(short value)	O
	writeString(java.lang.String value)	O
TemporaryQueue	delete()	O
TemporaryTopic	delete()	O

O: Supported

X: Not supported

Note 1) Synchronous and asynchronous messages cannot be received simultaneously by one MessageConsumer.

API List of the Package javax.jms (Part 7)

Table 17-9 API List of javax.jms

Interface name/class name	Method	Support
TextMessage	getText()	O
	setText(java.lang.String string)	O
Topic	getTopicName()	O
	toString()	O (Note 1)
TopicConnection	createConnectionConsumer(Topic topic, java.lang.String messageSelector, ServerSessionPool sessionPool, int maxMessages)	O
	createDurableConnectionConsumer(Topic topic, java.lang.String subscriptionName, java.lang.String messageSelector, ServerSessionPool sessionPool, int maxMessages)	O
	createTopicSession(boolean transacted, int acknowledgeMode)	O
TopicConnectionFactory	createTopicConnection()	O
	createTopicConnection(java.lang.String userName, java.lang.String password)	O (Note 2)
TopicPublisher	getTopic()	O
	publish(Message message)	O
	publish(Message message, int deliveryMode, int priority, long timeToLive)	O (Note 3)
	publish(Topic topic, Message message)	O
	publish(Topic topic, Message message, int deliveryMode, int priority, long timeToLive)	O (Note 3)
TopicRequestor	close()	O
	request(Message message)	O
TopicSession	createDurableSubscriber(Topic topic, java.lang.String name)	O
	createDurableSubscriber(Topic topic, java.lang.String name, java.lang.String messageSelector, boolean noLocal)	O
	createPublisher(Topic topic)	O
	createSubscriber(Topic topic)	O
	createSubscriber(Topic topic, java.lang.String messageSelector, boolean noLocal)	O
	createTemporaryTopic()	O

Interface name/class name	Method	Support
	createTopic(java.lang.String topicName)	O
	unsubscribe(java.lang.String name)	O
TopicSubscriber	getNoLocal()	O
	getTopic()	O

O: Supported

X: Not supported

Note 1) 'com.fujitsu.interstage.jms:<Topic name>::<the group name of the event channel>::<the channel name of the event channel>' is returned.

Note 2) userName and password are ignored.

Note 3) timeToLive supports in milliseconds. However, since the message timeout of the event channel is in seconds, timeToLive is rounded off to the nearest value.

API List of the Package javax.jms (Part 8)

Table 17-10 API List of javax.jms

Interface name/class name	Method	Support
XAConnection	No method	
XAConnectionFactory	No method	
XAQueueConnection	createQueueSession(boolean transacted, int acknowledgeMode)	X
	createXAQueueSession()	X
XAQueueConnection Factory	createXAQueueConnection()	X
	createXAQueueConnection(java.lang.String userName, java.lang.String password)	X
XAQueueSession	getQueueSession()	X
XASession	commit()	X
	getTransacted()	X
	getXAResource()	X
	rollback()	X
XATopicConnection	createTopicSession(boolean transacted, int acknowledgeMode)	X
	createXATopicSession()	X

Interface name/class name	Method	Support
XATopicConnection Factory	createXATopicConnection()	X
	createXATopicConnection(java.lang.String userName, java.lang.String password)	X
XATopicSession	getTopicSession()	X

O: Supported

X: Not supported

Note) The XA interface is not supported.

Part VI

Connector Edition

Chapter 18

Basic Functions of the Interstage Connector

This chapter explains the basic function of Interstage Connector.

Connection Management

Interstage connector provides a function to connect a resource of connector by acquiring ConnectionFactory of the connector from JNDI.

Interstage Connector's pool manager manages connection information after making the first connection. Consequently, it makes it possible to access the resource from many clients or to construct an application environment for which a frequent access to the resource access is necessary.

Timeout for the Pooled Connection

The container automatically releases unused connections that exceed the time-out period.

To set the time-out value refer to the following file. The initial value is 600 (= 10 min.). The unit is seconds.

Windows

C:\Interstage\J2EE\etc\JCA\jca.properties

Solaris OE Linux

/opt/FJSVj2ee/etc/jca/jca.properties

Set the following property in the above-mentioned file. The initial value (600) is set by default.

Property name	Property value
idle.resource.threshold	Timeout value

To set the time-out period to 300 seconds, change it as follows.

```
idle.resource.threshold=300
```

Transaction Management

Interstage connector provides functionality to manage the transaction of the resource by using the transaction function provided by EJB. The transaction across several resource managers can be managed this way.

Supported Transaction Support Level

Interstage Connector supports each transaction level of resource adapter.

The transaction support level of resource adapter is defined in the transaction-support tag of deployment descriptor. There is a difference at the transaction level supported by resource adapter according to the specified value as follows.

Table 18-1 Transaction Level Supported by Resource Adapter

Transaction type	Usage
XA transaction support	"XATransaction" is specified for transaction-support tag. The global transaction function of EJB is used, and the transaction management that cooperates with the resources other than resource adapter is possible. The transaction is processed by two-phase committing protocol.
Local transaction support	"LocalTransaction" is specified for local transaction support transaction-support tag. It cannot be used with two-phase committing protocol (2PC) unlike the XA transaction. Only one resource recommended to be accessed in one transaction because it is always processed by one phase committing protocol.
No transaction	"NoTransaction" is specified for transaction-support tag. Resource adapter to which this transaction type is supported does not cooperate with the transaction.

It is possible to cooperate with the transaction ("Container" is specified for a transaction attribute) that the container controls when resource adapter that supports XA transaction or a local transaction is used.

Note when Transaction Function is Used

To use XA transactions, do as follows: From the Interstage Management Console, select [WorkUnit] > [IJServer name] > [EJB container setting] > [Use distributed transaction] and select "Use."

Security Management

For Interstage Connector, a safe security management function to EIS is supported. Safety to EIS is secured by this function, and the resource that EIS manages is protected. This function uses the resource connection manager function of EJB.

Table 18-2 Security Management Function

How to sign on	Specification of resource connection	Operation
Sign-on by application management	Application	The resource is accessed by specifying a user ID/password by the application. Information set by the application has the following two cases. In case that connect from EJB application to resource, and who can connect to the resource are specified in the EJB application: Use user ID and password that are specified in EJB application. In other cases: Use user ID and the password to which the user is authorised by the J2EE application client or the Web applications.
Sign-on by container management	Container	This connects by user ID/password to which the container is set as follows: Interstage Management Console

Part VII

Tool Edition

Chapter 19

J2EE Resource Access Definition

The J2EE resource access definition is a function for creating a resource access definition that is needed for referencing JDBC, JMS, connector, or JavaMail objects by the naming service using GUI windows.

The J2EE resource access definition is used when the Interstage client function is installed. To create a resource definition on the server, use the Interstage Management Console.

This chapter explains how to activate the J2EE resource access definition.

Note

The J2EE resource access definition cannot be used to define PostgreSQL data sources. When using the client package to access PostgreSQL from the J2EE application client, use the fjj2eeadmin command.

Refer to the Reference Manual (Command Edition) for details of the fjj2eeadmin command.

Activating the J2EE resource access definition

This section explains the following topics:

- J2EE resource access definition activation command
- Initial window for J2EE resource access definition

J2EE resource access definition activation command

Use the j2eejndisetup command to activate the J2EE resource access definition.

```
j2eejndisetup
```

Refer to "JNDI Operation Commands" in the “Reference Manual (Command Edition)” for details of the j2eejndisetup command.

Refer to "Messages Output during Resource Access Definition" in the Messages for explanations of the error and warning messages displayed during execution of the J2EE resource access definition and the required actions.

Notes

- When executing the J2EE resource access definition, make sure the following values are set in the environment variables.

  [CLASSPATH]

  C:\Interstage\j2ee\lib\isj2ee.jar

  [PATH]

  C:\Interstage\bin

- A Java environment must be set up to execute the J2EE resource access definition.
  - Installing Java

    Install the Interstage Apworks client operation package.
  - Setting up the Java environment

    Make sure the following value is set in environment variable PATH.

    When the JDK1.3 system is used : C:\Interstage\JDK13\jre\bin

    When the JRE1.3 system is used : C:\Interstage\JRE13\bin

    When the JDK1.4 system is used : C:\Interstage\JDK14\jre\bin

    When the JRE1.4 system is used : C:\Interstage\JRE14\bin
- Execute the J2EE resource access definition with administrator authority.

Initial window for J2EE resource access definition

When the J2EE resource access definition is activated normally, the "J2EE resource access definition" window shown below opens.

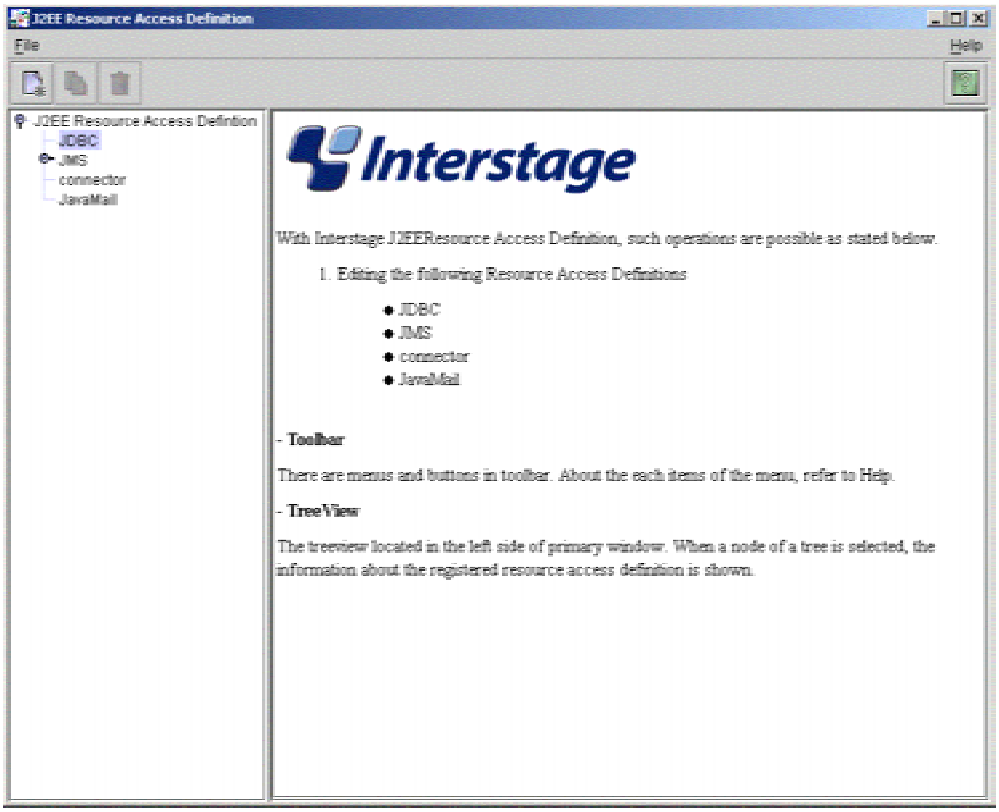


Figure 19-1 J2EE Resource Access Definition Window

Index